Better scoring schemes for the recognition of functional proteins by protomata

Manon Ruffini

Master research Internship

Internship report

Better scoring schemes for the recognition of functional proteins by protomata

Domain: Bioinformatics

Author:
Manon Ruffini

Supervisor:
François Coste
Dyliss

Abstract: Proteins perform very important functions within organisms. Predicting these functions is a major problem in biology. To address this issue, predictive models of functional families from the sequences of amino acids that form the proteins have been developed. The Dyliss team developed a machine learning algorithm, named `Protomata-learner`, that learns weighted automata representing these families and the possible disjunctions between members. New sequences can be compared to these models and assigned a score to predict their belonging to the family.
Despite good results, the sequence weighting strategy and the null-models in Protomata are rather basic. During my internship, I investigated alternative sequence weighting strategies and null-models. Besides, the expressivity of Protomata leads to a great variability of scores and the choice of the classification threshold was left to the user. So, I proposed a normalization of the score, and a method to assess the significance of scores, to simplify the prediction. I implemented these new strategies and compared them on several data sets. Preliminary results show a good improvement of the prediction power of the computed models.
Keywords: proteins, statistical modelling, automata, Dirichlet mixture, sequence weighting, null-model, significance

Contents

Acknowledgements

First, I would like to thank François Coste for the opportunity of this internship, his patience and pedagogy, and his encouragements that helped me gain some confidence. Thank you to Jérémie Bourdon for his interest in this work and helpful ideas. I would also like to thank Anne Siegel and the Dyliss team for their warm welcome. Finally, thank you to my fellow interns for their encouraging and communicative cheerfulness at all times.

1 Introduction

Proteins perform a major role within biological organisms. For example, they enable the transportation of molecules, the replication of DNA, the catalysis of metabolic reactions, or the response to stimuli. A protein is composed of a chain of amino acids, that folds into a three dimensional structure, determining a function. It can be seen as a word over the twenty-letter alphabet of amino acids. Although identifying the function of a protein is a costly and complicated task, the sequence of the amino acids forming the chain can be rather easily determined thanks to sequencing technologies. Therefore, the number of available sequences has been growing exponentially, along with the need for an automatic method for identifying their function.

To infer the function of a protein from its sequence of amino acids, a major approach consists in training statistical models of functional families from available sequences, and comparing new sequences to these models. State-of-the-art methods in bioinformatics, like profile hidden Markov models (pHMM), proceed with identifying and modeling conserved segments in available sequences from a given family of proteins. The model is then used to compute a score for any previously unseen sequence, that is used to decide whether or not the corresponding protein belongs to the family. However, proteins performing the same function might come from different sub-families that do not share the same segment conservations, and current state-of-the art models, as pHMMs, fail at modeling these differences. To tackle this issue, the Dyliss team introduced a more expressive model, called Protomata, that enables the representation of these disjunctions.

Although Protomata computes more expressive models, it only achieves similar results than other state-of-the-art statistical modeling strategies for families of proteins. This might come from the fact that Protomata mostly relies on strategies that were originally designed for pHMMs, and these approaches might need to be adapted to the characteristics of the new models. Moreover, once a sequence is scored, no proper method has been defined to decide if it should be accepted or rejected.

During my internship, I investigated the existing weighting strategies for training sequences, developed for pHMMs, and adapted some of them to Protomata, in order to achieve a more accurate and less biased model of the family of proteins. Moreover, I suggested an enhancement of the scoring strategy through a normalization of scores and a reduction of the space of possibly parsed sequences. In addition, I proposed an empirical approximation of the distribution of the scores, in order to achieve an estimate of the corresponding significance, indicating whether it should be accepted or rejected.

In section 2, the main statistical models for families of proteins are presented, along with the new approach introduced with Protomata. Then, in section 3, possible improvements in the preprocessing of the training data, and in the scoring strategy are suggested. Next, in section 4, the distribution of scores are explored, in order to achieve a significance for the score of any new parsed

sequence, that should enhance the predictive power of the model. Previously introduced strategies were tested on several data sets: experimental details and preliminary results are summarized in section 5.

2 State of the art

Given sequences from a family of related proteins, we aim at building a model of the family. We want this model to be explicit, so that experts might extract new knowledge from it, and to be able to find new members of the family, in order to label previously unseen proteins.

In subsection 2.1, we will introduce position specific modeling methods, that rely on the alignment of the sequences of the family being modeled. Some strategies used to enhance the estimation of the parameters of these models will be described. Then, in subsection 2.2, we will introduce a new approach, called Protomata, introduced by the Dyliss team, that relies on a new kind of multiple alignment, that takes into account the possible disjunctions within the family of proteins being modeled.

2.1 Position specific modeling

Before going any further, we need to properly define the families of proteins that we will consider, as there are many possibilities of grouping proteins into families. We can consider functional or structural families, defined as families of proteins that share the same function or structure. In bioinformatics, we usually focus on proteins deriving from a shared common ancestor, called homologous, that are likely to have kept the same function or structure due to natural selection. They are called equivalogous when they are homologous and have kept the same function through evolution.

2.1.1 Sequence alignment

The family models presented here are build from an alignment of the available sequences, introduced hereafter. A sequence alignment is based on a measure of similarity of the sequences. This measure of similarity can be described by a substitution matrix, defined below.

Substitution matrices Due to selection pressure, amino acids are more likely to be substituted by residues with close physical and chemical properties, that will not affect the structure or the function of the protein. Sequences that share common history can share very conserved segments in their sequences, from which we can deduce the probability of each amino acid to be substituted by another. These probabilities are used to compute a 20×20 substitution matrix, such as the BLOSUM62 matrix ([Coste, 2016]). Such matrices provide measures of similarity between sequences: if two sequences have very substitutable amino acids in the same positions, then they are similar.

Sequence alignment The proteins that we consider have a common evolution history, and hence their sequences share similarities, that can be harnessed to determine characterizations. Given a set of homologous protein sequences, we align the conserved positions of the sequences in columns, where each column is assigned a position in each sequence, ideally corresponding to the evolution of the position in the offspring sequence. Gaps can be introduced, in order to maintain the alignment,

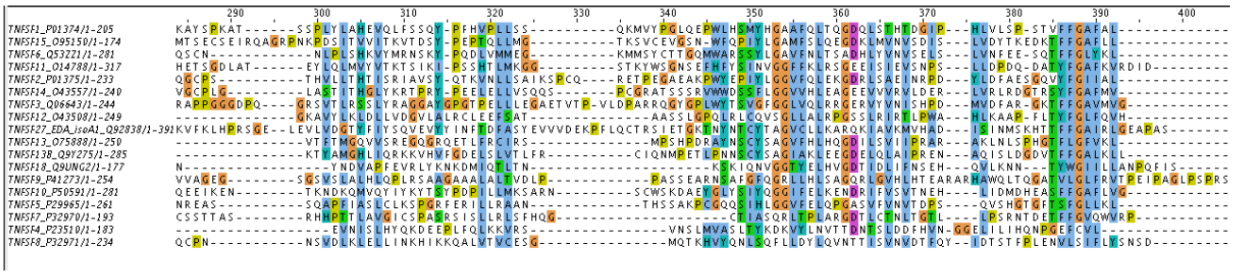


Figure 1: Alignment of sequences from the family of tumor necrosis factors, performed with ClustalW, reprinted from [Kerbellec, 2008]

even if sequences present extra amino acids, due to insertions and deletions, that might have happened in the evolution process. The evolutionarily correct alignment is very difficult to infer, so it is approximated by the alignment that maximizes the similarity score, computed from a substitution matrix ([Kerbellec, 2008]). Programs that compute sequence alignments are available, like ClustalW ([Thompson et al., 1994]). An example of a multiple alignment is given in figure 1. A sequence is said to be involved in a column if its corresponding aligned position presents an amino acid, and not a gap.

The longer the sequences, the harder it is to find an appropriate alignment. Mostly, sequences present very well conserved regions, that are important to the family, and unspecific chains of amino acids in between. Hence, instead of computing the global alignment with the whole collection of sequences, algorithms like MEME or DIALIGN ([Grundy et al., 1997],[Coste, 2016]) were developed to compute a local alignment, that only involves a small segment of the sequences. If several regions can be locally aligned, we will talk about a multiple local alignment.

2.1.2 From sequence alignment to family modeling

The alignment of sequences of homologous proteins shows the conservations in the family. Moreover, it can be used to build the following predictive models, that are employed to score previously unseen sequences and determine whether they belong to the family or not.

Position specific scoring matrix (PSSM) Position specific scoring matrices (PSSM) describe the successive positions of the sequences of the family ([Coste, 2016]). Each column of the matrix corresponds to a column in the alignment, and each line to an amino acid. The coefficient of the line a and column i corresponds to the probability of finding amino acid a at the i-th position, that can be estimated by maximum likelihood. Let us denote $p_i(a)$ this probability, and $o_i(a)$ the number of occurrences of a at position i, then:

$$p_i(a) = \frac{o_i(a)}{\sum_{a'=1}^{20} o_i(a')} \tag{1}$$

For a given new sequence, its positions are aligned with the columns of the PSSM, and the probabilities of each position are combined, leading to the probability of the alignment of the sequence to the PSSM. This probability is then compared to a threshold, in order to decide of the

belonging of the protein to the family. A sketch of the typical topology of a PSSM is shown in figure 2.

This topology is suited to represent very conserved and rather small regions with few insertions or deletions.

Hidden Markov model profile (pHMM) In order to represent longer sequences, the model must be more flexible. Specifically, insertions and deletions must be taken into account to be in harmony with the evolution process.

Hidden Markov model profiles (pHMMs) can be seen as PSSMs with possibilities of insertion and deletion added to each column ([Coste, 2016]). Their topology is a left-to-right hidden Markov model (HMM) with three hidden states for each position: a state of match, that corresponds to the column in the PSSM, a state of insertion, and a state of deletion (figure 2). This model allows small and rare insertions and deletions, but also the insertion of greater segments of amino acids.

Once this hidden Markov model profile (pHMM) is computed, the best alignment of a new sequence can be computed, along with its probability, using the Viterbi algorithm for example.

The construction of a pHMM usually starts from a multiple local alignment of the sequences. The columns of the PSSM that involve enough sequences, say more than half of the set for instance, are converted in match states, and associated with the corresponding insertion and deletion states. The coefficients in the match states, called emission probabilities, can be computed as in the columns of PSSMs. As far as transition probabilities are concerned, they are assigned as follows: for each pair of distinct states (k, l), let us denote the transition probability t_{kl} and frequency T_{kl}, by the application of the maximum likelihood principle, we get the following formula ([Kerbellec, 2008]):

$$t_{kl} = \frac{T_{kl}}{\sum\limits_{l'} T_{kl'}} \tag{2}$$

The amino acids that are generated by the insertion states are not characteristic of the family. However, their number is important, since it determines the relative position of conserved segments in the sequence. Hence, the quantity of inserted amino acids between two conserved segments, i.e. the number of loops on the corresponding insertion state, has to be controlled.

Let us consider an insertion state i, let g_i be the probability of a loop on i and μ_i be the average number of inserted amino acids (computed from the data). Let X be the random variable that represents the number of loops on i. Then, for all $n \in \mathbb{N}$:

$$\mathbb{P}(X = n) = g_i^n(1 - g_i) \text{ and } \mathbb{E}[X] = \frac{g_i}{1 - g_i} \tag{3}$$

The parameter g_i is then set, so that $\mathbb{E}[X] = \mu_i$:

$$g_i = \frac{\mu_i}{1 + \mu_i} \tag{4}$$

Meta-MEME Though pHMMs enable the representation of longer sequences, they induce the estimation of a lot of parameters. Moreover, in sequences, the succession of some positions in very conserved regions might be important, and no insertion or deletion should be allowed. For these reasons, pHMMs can be simplified to Meta-MEME models ([Grundy et al., 1997]): insertion and

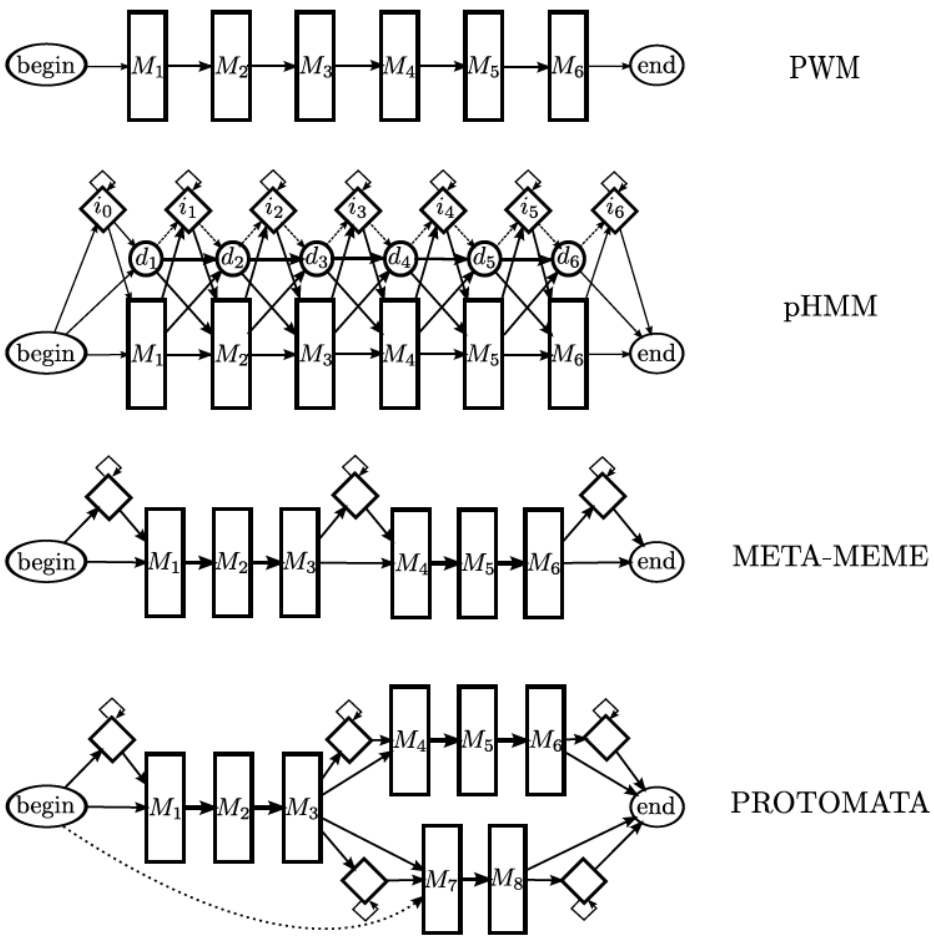


Figure 2: PSSM (PWM), pHMM, Meta-MEME and Protomata types of architecture, reprinted from [Coste, 2016]

deletion states are removed, some of the match states are kept and grouped in blocs, that correspond to the most characteristic segments of the family. These blocs are separated with gaps, that enable the insertion of amino acids (figure 2). Blocs can be assimilated to PSSMs, and gaps to insertion states. Parameters are estimated as before.

2.1.3 Pseudo-counts on emission probabilities

We have described several strategies for modeling families of proteins. Doing so, we limited ourselves to the maximum likelihood principle in the estimation of emission and transition probabilities. However, this approximation can be greatly improved. Indeed, as only few sequences are available for the construction of the model, background knowledge needs to be integrated to avoid overspecialization.

Specifically, as the number of parameters to be estimated, in a pHMM for example, can be high compared to the number of sequences usually available, some of the previous emission probabilities

might be estimated as zero. However, the corresponding amino acids might actually appear in other sequences of the family. Typically, if we have less than twenty learning sequences, it is impossible for each amino acid to appear, since there are twenty of them. However, if a new sequence happen to present a previously unseen letter in a position, and a lot of similarity otherwise, we don't want its probability to be zero, since we need to recognize new members of the family. To tackle this issue, we can add pseudo-counts to the observed frequencies. The intuition is that we pretend we observed every amino acid even if we didn't, especially in the case of a small training set. If there are a lot of sequences for the training phase however, and an amino acid never appears in a position, we are more willing to believe that it is due to a specificity of the family, and not a lack of data, and pseudo-counts should become insignificant compared to the data contributions.

A first possibility is to add a constant to all the counts, for example:

$$p_i(a) = \frac{o_i(a) + 1}{\sum_{a'}(o_i(a') + 1)} \tag{5}$$

A more elaborated constant could take into account the background probability of each amino acid a, denoted q_a. From the numerous sequences, the share of each amino acid can be assessed and employed: an amino acid that usually appears more often is more likely to be observed at any position:

$$p_i(a) = \frac{o_i(a) + Aq_a}{\sum_{a'} o_i(a') + A} \tag{6}$$

The constant A is to be chosen as the weight to put on pseudo-counts. It appears that a value around twenty, corresponding to the number of amino acids, seems to provide satisfying results [Kerbellec, 2008].

If there are only a few training sequences, we assume that the prior knowledge remains valid, and the emission probability is close to the background probability. However, if a lot of proteins are available, it seems reasonable to believe the data, and the pseudo-counts get dominated by the observed counts.

The next step in the elaboration of pseudo-counts consists in considering the observed amino acids at the given position and deduce from them the other possible residues, depending on the fact that some amino acids are more likely to be substituted than others, due to their physical and chemical properties.

Substitution matrices As explained above, substitution matrices are computed from sets of conserved sequences and provide information about the substitutability of amino acid residues. This expertise can be injected in the calculation of the emission probabilities: if the residues in the new sequence are very likely to have been substituted from the observed amino acids in the family, then the sequence should get a high score ([Henikoff and Henikoff, 1996]).

Let us denote $S = (s_{ab})_{1 \leqslant a,b \leqslant 20}$ the substitution probabilities computed from a substitution matrix. Then for all amino acids a and b, $s_{ab} = s_{ba}$ correspond to the probability of a and b being substituted. The emission probabilities can be adjusted as follows, for each position i and each

amino acid a:

$$p_i(a) = \sum_{b=1}^{20} \frac{o_i(b)}{\sum_{b'} o_i(b')} s_{ab} \tag{7}$$

This method embeds more precise knowledge in the estimations, and has been proven to be efficient ([Henikoff and Henikoff, 1996]). However, it presents two drawbacks. First, the number of training sequences is not fully taken into account. Prior knowledge is integrated, regardless of the fact that it is very needed in the case of a small training set, but should become less important when more data is available. Secondly, the substitutability of amino acids is computed as a mean over all possible cases. Whether a substitution happens because of a property or another makes no difference.

Dirichlet mixtures Pseudo-counts based on Dirichlet mixtures enable the addition of more precise prior knowledge in the estimation of parameters. The idea consists in describing the prototypical distributions of the columns in the alignments ([Sjolander et al., 1996]). Unlike substitution matrices, we aim at taking into account the important properties of amino acids at each position in order to determine the more substitutable residues. For example, if a column of the alignment presents only small amino acids, Dirichlet mixtures will increase the emission probabilities of other small amino acids, but not the probability of other residues, even if they are exchangeable with the observations in other cases.

The pseudo-count method based on Dirichlet priors is described below. It consists in computing typical column distributions ; assigning a probability to these distributions for each column of the alignment, depending on the observations ; and correcting the emission probabilities by adding the possibility of unobserved amino acids, that are consistent with the probability distribution.

Background on Dirichlet densities and Dirichlet mixtures Let us consider an alignment of protein sequences and a column of amino acids corresponding to a position of the alignment. Let us denote n the size of the columns. For each line in the column, there are 20 possible outcomes : the 20 amino acids, with corresponding probabilities $(p_a)_{a\in[\![1,20]\!]}$. If the random variables $o(a)$ indicate the number of times the amino acid a is observed in the column, then the vector $\vec{o} = (o(a))_{a\in[\![1,20]\!]}$ follows a multinomial distribution:

$$\mathbb{P}(o(1) = n_1, \ldots, o(20) = n_{20}) = \begin{cases} \frac{n!}{n_1!\ldots n_{20}!} p_1^{n_1} \ldots p_{20}^{n_{20}} & \text{when } \sum_{a=1}^{20} n_a = n \\ 0 & \text{otherwise} \end{cases} \tag{8}$$

So, we will consider that a column in a sequence alignment corresponds to a multinomial distribution. The modeling strategy now consists in estimating the distribution for each column, depending on the observations. A pseudo-count method consists in describing the distribution of the columns from all alignments in the multinomial space. Emission probabilities are then adjusted thanks to this distribution.

We recall that a Dirichlet density ρ of parameters $\vec{\alpha} = (\alpha_a)_{a\in[\![1,20]\!]}$, with $\alpha_a > 0$, pour tout a, is a density over the set of probability vectors $\mathcal{P}_{20} = \left\{\vec{p} = (p_a)_{a\in[\![1,20]\!]} / \forall a \in [\![1,20]\!], p_a \geqslant 0, \sum_{a=1}^{20} p_a = 1\right\}$,

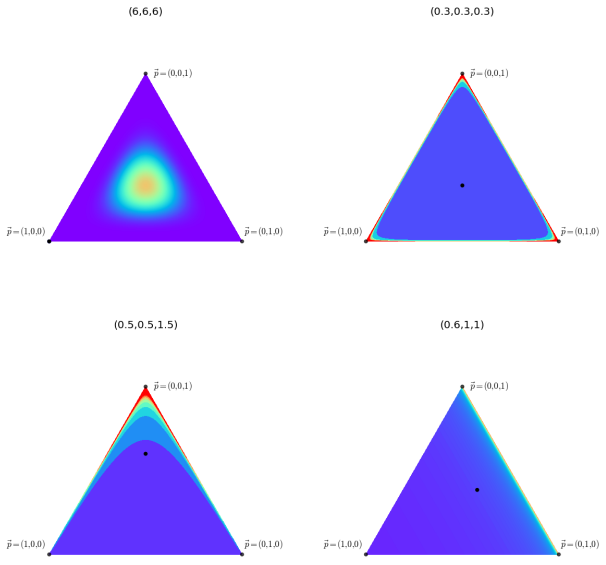


Figure 3: Density plots for four Dirichlet distributions on $\mathcal{P}_3$. Darker blue corresponds to lower probability density and red to higher density. **Top left:** Density of parameters $(6, 6, 6)$. The probability density is concentrated around the expected probability vector $(\frac{1}{3}, \frac{1}{3}, \frac{1}{3})$. **Top right:** Density of parameters $(0.3, 0.3, 0.3)$. The expected probability density is again $(\frac{1}{3}, \frac{1}{3}, \frac{1}{3})$, but the parameters are lower than 1, so the density is concentrated near the edges: it favors sparse multinomials. **Bottom left and right:** Densities of parameters $(0.5, 0.5, 1.5)$ and $(0.6, 1, 1)$ respectively that favor sparse multinomials. These distributions are asymmetric

defined as follows [Sjolander et al., 1996]:

$$\rho(\vec{p}) = \frac{1}{Z} \prod_{a=1}^{20} p_a^{\alpha_a - 1} \tag{9}$$

where $Z = \frac{\prod_{a=1}^{20} \Gamma(\alpha_a)}{\Gamma(|\vec{\alpha}|)}$ is a constant enabling $\int_{\mathcal{P}} \rho(\vec{p}) d\vec{p} = 1$.

The parameter $|\vec{\alpha}| = \sum \alpha$ is called the concentration parameter. The mean vector of the distribution is given by $\vec{p} = \frac{\vec{\alpha}}{|\vec{\alpha}|}$. So, the distribution is also characterized with $(\vec{p}, \alpha)$ ([Altschul et al., 2010]). Figure 3 shows the influence of the parameters, in the simple case of three letters, i.e. on $\mathcal{P}_3$. We can see that the higher the concentration parameter $|\vec{\alpha}|$, the more concentrated the distribution, around the expected probability vector. In the extreme case where the α_a are lower than 1, the density favors probability vectors that provide a non-negligible probability to only one letter, or at least to a small number of letters.

As far as proteins are concerned, the multinomial space presents several regions with high prior probabilities, and a single Dirichlet distribution would fail at correctly representing all these regions. Hence, the approach can be generalized and a mixture of several densities can be considered. One density corresponds to a densely populated region of the multinomial space, and a mixture is a linear combination of these densities.

Given M Dirichlet densities $(\rho_j)_{1 \leqslant j \leqslant M}$ and M positive mixture coefficients $q_1, \ldots, q_M$ such that $\sum_{j=1}^{M} q_j = 1$, a Dirichlet mixture ρ of components $\rho_1, \ldots, \rho_M$ is defined as:

$$\rho = q_1 \rho_1 + \cdots + q_M \rho_M \tag{10}$$

The set of parameters of the mixture density will be denoted $\Theta = (\vec{\alpha}_1, \ldots, \vec{\alpha}_M, q_1, \ldots, q_M)$.

Inference of Dirichlet priors From the large amount of sequence alignments that are available, a data set can be created with the columns of these alignments, and used to infer prior knowledge about proteins and amino acids. A Dirichlet mixture is seeked, in order to represent at best the distribution of the alignment columns in the multinomial space. To that end, the Dirichlet mixture is chosen as the one that maximizes the probability of the columns. The number of components has yet to be chosen. Not enough components would not enable an accurate representation of the set of columns, but too many would lead to the over-fitting of the distribution to the data. The most used mixtures present between 9 and 30 components. A more precise description of the inference of Dirichlet priors is given in Appendix 6.

Using Dirichlet priors in the estimation of emission probabilities Once a Dirichlet mixture is computed from the data set of available alignments, it can be used to enhance the estimation of emission probabilities.

First, let us consider the case of a single Dirichlet component of parameters $\vec{\alpha}$. Given a position i in an alignment and its corresponding count vector $\vec{o_i} = (o_i(a))_{1 \leqslant a \leqslant 20}$, then the emission probability of an amino acid a is given by ([Sjolander et al., 1996], [Altschul et al., 2010]):

$$p_i(a) := \mathbb{P}(a|\vec{\alpha}, \vec{o_i}) = \frac{o_i(a) + \alpha_a}{|\vec{o_i}| + |\vec{\alpha}|} \tag{11}$$

This comes from Bayes' theorem, that proves that the posterior distribution after the observation of a letter b is also a Dirichlet distribution, with parameters $\vec{\alpha}'$, where $\alpha'_a = \alpha_a$ if $a \neq b$ and $\alpha'_b = \alpha_b + 1$.

When the Dirichlet mixture has more than one component, the law of total probability gives:

$$p_i(a) = \sum_{j=1}^{M} \mathbb{P}(\vec{\alpha}_j|\vec{o_i}, \Theta)\mathbb{P}(a|\vec{\alpha_j}, \vec{o_i}) = \sum_{j=1}^{M} \mathbb{P}(\vec{\alpha}_j|\vec{o_i}, \Theta)\frac{o_i(a) + \alpha_j(a)}{|\vec{o_i}| + |\vec{\alpha}_j|} \tag{12}$$

where $\vec{\alpha}_j = (\alpha_j(a))_a$ denotes the parameters of the j-th component of the mixture. Moreover, the application of Bayes' rule and the law of total probability gives:

$$\mathbb{P}(\vec{\alpha}_j|\vec{o_i}, \Theta) = \frac{q_j\mathbb{P}(\vec{o_i}|\vec{\alpha}_j)}{\sum_{k=1}^{M} q_k\mathbb{P}(\vec{o_i}|\vec{\alpha}_k)} \tag{13}$$

In [Sjolander et al., 1996], it is stated that the probability of the vector of counts equals the sum of the probabilities of all the distinct observations that lead to this count vector, leading to:

$$\mathbb{P}(\vec{o_i}|\vec{\alpha}_k) = \frac{\Gamma(|\vec{o_i}| + 1)\Gamma(|\vec{\alpha}_k|)}{\Gamma(|\vec{o_i}| + |\vec{\alpha}_k|)} \prod_{a=1}^{20} \frac{\Gamma(o_i(a) + \alpha_k(a))}{\Gamma(o_i(a) + 1)\Gamma(\alpha_k(a))} \tag{14}$$

From equation 12, we can see that the quantity $\sum_{j=1}^{M} \mathbb{P}(\vec{\alpha}_j|\vec{o_i}, \Theta)|\vec{\alpha_j}|$ corresponds to the additional number of sequences we "pretend to see" in the elaboration of pseudo-counts.

An alternative formula is also used in the litterature. Indeed, in [Ye et al., 2011], it is argued that all the observations that lead to the same vector of counts are equiprobable, thanks to a property of exchangeability of Dirichlet mixtures ([Altschul et al., 2010]). So, instead of considering

the probability of the vector of counts, they consider the probability of a single observation that leads to said vector:

$$\mathbb{P}(\vec{o_i}|\vec{\alpha}_k) = \frac{\Gamma(|\vec{\alpha}_k|)}{\Gamma(|\vec{o_i}| + |\vec{\alpha}_k|)} \prod_{a=1}^{20} \frac{\Gamma(o_i(a) + \alpha_k(a))}{\Gamma(\alpha_k(a))} \tag{15}$$

One can note (from equation 12) that if there are only a few observations, the emission probabilities are estimated almost only through prior knowledge, whereas when the number of training sequences becomes significant, the counts get dominant over the pseudo-counts, and the estimation tends towards the maximum-likelihood solution.

Pseudo-counts based on Dirichlet mixtures enable the integration of precise prior knowledge on protein sequences. This usage of background knowledge provides the possibility of modeling accurately families of proteins based on fewer and fewer sequences.

In practice, PSSMs are rather employed for modeling short DNA transcription factors. As far as proteins are concerned, the current state-of-the-art techniques mostly rely on pHMMs, for which several programs are available, like HMMER or SAM. These programs default sets of parameters include pseudo-counts based on Dirichlet mixtures ([Eddy and Wheeler, 2015], [Hughey et al., 2003]).

2.2 Protomata

Previously described state-of-the-art modeling strategies rely on the assumption that sequences share enough similarities to be all aligned, at least locally. However, in the evolution process, disjunctions might be observed. Indeed, some sub-families might have evolved independently of one another, and there might be some regions that are common to some sequences, but not all of them. Moreover, when it comes to functional families of non-necessarily homologous proteins, some sub-families might share very few similarities.

In order to get a more accurate representation of a family of proteins, the Dyliss team recently introduced a new model, called Protomata. This new approach consists in modeling the conserved regions in the sequences, that do not necessarily involve all the proteins, and describing the possible successions of these conservations. Dyliss created a program, named Protomata-Learner, that computes a new kind of multiple sequence alignment, said partial and local, from the set of sequences, and infers the Protomaton from it.

2.2.1 Partial Local Multiple Alignment (PLMA)

The Meta-MEME model described above was based on local alignments: the alignments only involved small segments of the sequences to enhance the conservation. In order to keep improving the detection of pertinent conservations, partial local alignments (PLA) can be considered: they are local alignments that do not necessarily involve all the sequences ([Coste and Kerbellec, 2006], [Kerbellec, 2008]). The set of compatible PLAs constitutes a partial local multiple alignment (PLMA) (figure 4).

An algorithm to compute a PLMA is described in [Coste and Kerbellec, 2006]. The authors proceed by first detecting similar fragment pairs (SFPs), and ordering them according to their support in the set of sequences. These sorted SFPs are then considered in turn to search for the best PLA including them, with an algorithm based on cliques or connected components, record it and discard the SFPs that are incompatible with it. Incompatibility appears when a position in

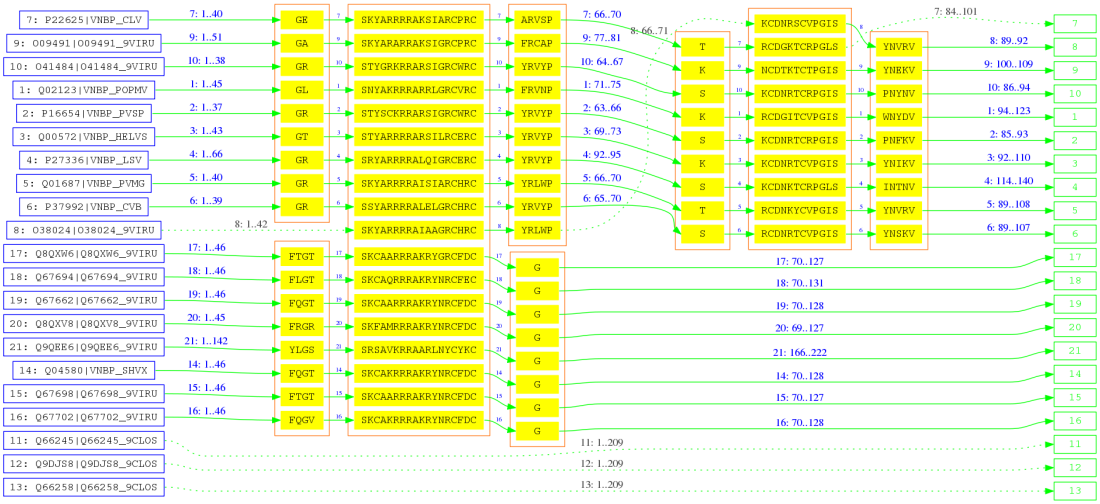


Figure 4: Example of a partial local multiple alignment (PLMA) computed with Protomata Learner. Shared segments are displayed in yellow and the residues involved are visible. The red square borders define the PLAs.

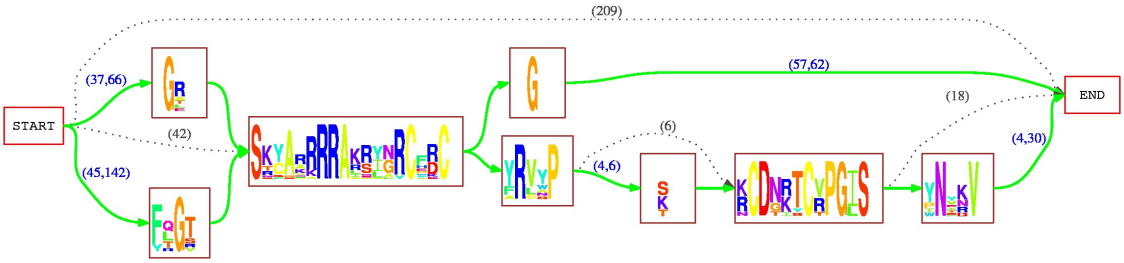


Figure 5: Example of a Protomaton computed with Protomata-Learner

a sequence corresponds through different SFPs to two different positions in another sequence. At the end of the iterations, a PLMA made of all the recorded PLAs is returned.

Given a partial local multiple alignment of the set of sequences, the Dyliss team proposed to build an automaton modeling the possible successions of PLAs ([Coste and Kerbellec, 2006]), represented each with a position specific scoring matrix (PSSM), linked by gap states, to enable the insertion of less important segments between the conserved regions.

More specifically, to infer the automaton from the training data, the sequences are firstly used to produce the maximal canonical automaton (MCA). The current states of the MCA correspond to the positions in each sequence, so a PLA can be merged, by merging the states of its local alignments. PLAs of the ordered list are merged, if they are compatible with the previous mergings (see [Coste and Kerbellec, 2006]).

Eventually, states that were not involved in the previous steps can be considered as not representative of the family, and treated as "gaps". An example of a Protomata is given in figure 5. As desired, the model is explicit, and the choice of the parameters should enable generalization and the discovery of new members of the family.

2.2.2 Estimation of parameters and scoring

From a training set of sequences, we are able to infer the topology of a protomaton that represents the family. The parameters of the model are yet to be determined.

So far, emission probabilities in match states are estimated and smoothed with Dirichlet mixtures, as described in formula 12. Every transition between blocs includes a gap state with a loop, that enables the insertion of amino acids, with their background probabilities. Unlike pHMMs, transitions are not assigned any probability (i.e. they are considered to be equiprobable), and there are no probabilities on loops over the gap states either.

Given a protomaton $\mathcal{M}$ and a sequence s to parse, its best alignment is computed thanks to the Viterbi algorithm ([Rabiner, 1989]), leading to the probability of said alignment. This probability corresponds to the product of the probabilities of each amino acid in the state it is aligned to, i.e. the background probability if it is aligned to a gap, or the corresponding emission probability if it is aligned to a match state.

Then, several scores are possible. We can either consider the raw probability of the sequence given the model $\mathbb{P}(s|\mathcal{M})$, or the log-probability $\log \mathbb{P}(s|\mathcal{M})$, or we can compare it to the probability of the sequence, assuming it was generated with the random model $\mathcal{R}$ (positions are modeled independently, each amino acid has its background probability at each position). This leads to the following log-likelihood ratio score:

$$\text{Score}(s) = \log \frac{\mathbb{P}(s|\mathcal{M})}{\mathbb{P}(s|\mathcal{R})} \tag{16}$$

Note that as the probabilities on gap states correspond to the probabilities of the random distribution, the log-likelihood ratio contribution of an amino acid a aligned with a gap state is equal to $\log \frac{q_a}{q_a} = 0$. So, the log-likelihood ratio score only depends on the alignment of the sequences to the match states:

$$\text{Score}(s) = \sum_i \log \frac{p_i(a_i)}{q_{a_i}} \tag{17}$$

where i covers the match states the sequence is aligned with and a_i denotes the residue of s aligned with i.

For any new sequence, Protomata returns its log-likelihood ratio score. If the score is positive, it means that the probability of the sequence given the model is higher than its probability given the random distribution. The most probable alignment leads to the best score for the sequence. Once it is computed, the score can be compared to a threshold in order to decide whether or not the sequence belongs to the family of proteins.

So, the new approach introduced with Protomata enables a more expressive representation of families of proteins, compared to other models, like pHMMs. The program designs an automaton modeling conserved regions and gaps, that indicates the allowed successions of conserved blocs. The presented scoring scheme was developed for pHMMs and also provides satisfying results with Protomata.

3 Enhancement of the scoring for Protomata

As stated before, Protomata relies on several strategies that were originally developed for pHMMs. Although these strategies provide satisfying results, several issues are raised, mostly because the

multiplicity of paths offered by Protomata leads to a great variability of scores in the parsing of new sequences. These issues are discussed hereafter.

Variability of the number of sequences involved in the PLAs Firstly, the sequences available do not necessarily represent well the family of proteins, and some subfamilies might be over-represented. In the computation of the PLMA, this might lead to blocs of very conserved regions, that involved a lot of sequences, that are very similar. In these blocs, the emission probabilities will be overspecialized compared to other PLAs, because more sequences, and thus more counts are observed, that will dominate the pseudo-counts. However, the amount of observation is not necessarily increased: two identical sequences do not provide more information than one.

However, if a PLA involves several sequences that present a very conserved segment and that are very dissimilar otherwise, then it is very likely that this segment is characteristic and the pseudo-counts should not overcome the observations. Besides, there is a balance to be found in the computation of the emission probabilities, between the actual observations $\vec{o}_i$ and the pseudo-counts $\vec{\alpha}_j$ (see equation 12).

So, there seems to be a need of an accurate weighting of the training sequences, in order to avoid over-specialization towards a sub-family, and to achieve a balance between observed counts and pseudo-counts.

Random distribution/Null-model selection Given a new sequence to be aligned with a protomaton, a log-likelihood ratio score is computed (see formula 16). The ratio enables the comparison of the probability of the sequence given the model of the family and its probability given the random distribution, that we will call null-model.

Note that if the parsed sequence has a composition of amino acids that is very different from the available sequences of the family, it is very unlikely to belong to the family. We want trained models to enable the rejection of sequences that do not belong to the family, but that are very similar to its sequences. Therefore, it could be interesting to consider other null-models, in order to compare the probability of the sequence given the model, and the probability of the sequence given a model of non-family members, that are similar to actual family members.

Variability of the scores depending on the path Let us note first that as we are considering a log-likelihood ratio scoring strategy, gaps do not contribute to the score of a parsed sequence, and the positive or negative contributions only correspond to match states. However, depending on the path, the number of match states can be very different. Therefore, if two sequences are aligned with different paths, their respective score might be very different.

For example, three different paths are represented in figure 6. A sequence aligned with the path 1 encounters 45 match states, whereas a sequence aligned with path 2 encounters 21 match states, and a sequence aligned with the complete exception on path 3 meets no match state at all. Therefore, sequences aligned with path 1 are more likely to get a high score compared to sequences aligned with paths 2 or 3.

Most importantly, a sequence very well aligned with a small path (i.e. with very few match states) and another sequence rather poorly aligned with a long path might get the same score in our model. Thus, the scores of sequences that are not aligned to the same paths can not be compared. Moreover, this raises the question of the choice of a threshold, to decide of the belonging to the

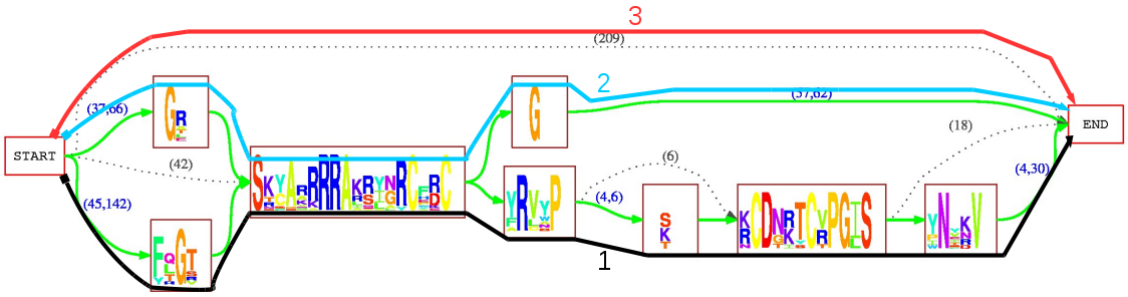


Figure 6: Example of a Protomaton computed with Protomata-Learner, with three possible alignments for new parsed sequences

family.

So, in order to benefit from the expressiveness enabled by Protomata, the scoring strategy needs to be adjusted, to improve the accuracy of the scores of parsed sequences, reduce their variability and enable their exploitation to classify sequences as belonging or not to the family.

In subsection 3.1, we will present different sequence weighting strategies, that aim at avoiding an over-representation of a sub-family and achieving a balance between observations and pseudo-counts. Then, in subsection 3.2, we will introduce different null-models, that should lead to a more relevant random distribution in the computation of the log-likelihood ratio. Finally, in subsection 3.3, we will try to reduce the variability of scores depending on the path by normalizing the score by the number of match states.

3.1 Weighting training sequences

The training set of sequences does not necessarily represent well the family of proteins, and some subfamilies might be over-represented. Moreover, in the estimation of the model, sequences are assumed to be independent, even though sequences are homologous and thus dependent. To compensate for this bias, weights can be attributed to protein sequences. We have investigated several strategies to achieve this.

Let us introduce the notations first. Assuming the weights $\omega_1, \ldots, \omega_N$ corresponding to the sequences $S_1, \ldots, S_N$ are computed, the number of occurrences of amino acid a in position i, $o_i(a)$ will be adjusted as follows :

$$o_i(a) = \sum_{j=1}^{N} \omega_j \delta(i, a, j) \tag{18}$$

where $\delta(i, a, j) = 1$ if the j-th sequence S_j presents an a at the i-th position and 0 otherwise.

There are two steps in the weighting of sequences. Internal weights have to be set, in order to assign smaller weights to similar sequences, and greater weights to rare ones. Also, external weighting has to be performed, to adjust the absolute weight of sequences, and be able to add the correct amount of pseudo-counts (the more observations, the less pseudo-counts, see equation 12).

Up to now in Protomata, internal weights are assigned using the tree-based approach performed by ClustalW, described below. Then, external weights are assigned as follows: internal weights are normalized so that the highest sequence weight is equal to one. This normalization of weights seems

rather arbitrary. Moreover, if the training set consists in two identical sequences, they will be both assigned a weight of one, whereas they provide as much information as one sequence. So, we have looked for more reasoned and accurate approaches.

Internal weighting approaches To avoid specialization towards possibly over-represented sub-families, we would like to attribute smaller weights to similar sequences, and higher weights to less represented ones. Several strategies are possible.

Tree-based approaches As far as families of homologous proteins are concerned, sequences can be related by an evolutionary tree. A weighting strategy consists in reconstructing this tree, and use it to weight the sequences : the ones that have recently diverged get a smaller weight than the more remote ones. Several tree-based weighting approaches have been introduced (see [Durbin et al., 1998] for more details).

In some cases though, for example when dealing with families of functional proteins, computing a tree relating the sequences might be costly, or not accurate enough. So, other weighting strategies were introduced, as described hereafter.

Distance-based approaches Protein sequences can be mapped in the "sequence space", and weights can be assigned depending on the distribution of the family in the sequence space: sequences located in a densely populated region are attributed a smaller weight than sequences located in a sparse region.

Voronoi weights ([Durbin et al., 1998]) are based on this idea : each sequence is attributed a weight proportional to the volume of empty space around it, i.e. the volume of its Voronoi cell within the sequence space. As we are dealing with a high-dimensional, a sampling strategy is used to avoid computing the Voronoi diagram.

Another strategy based on this idea was presented in [Henikoff and Henikoff, 1992]. Two aligned sequences that present more than a fixed percentage of identical residues are clustered. Then, each cluster counts as a single sequence and its weight is equally allocated between its sequences.

Position-based weighting strategy Previous weighting strategies are based on the entire protein sequences. However, the modeling strategies at stake here are position-specific, meaning that each position is considered independently. A weighting approach based on the diversity of sequences observed at each position in an alignment of sequences has been suggested in [Henikoff and Henikoff, 1994].

Given a column of an alignment of the sequences, each different amino acid (gap characters are treated as a 21st amino acid) is assigned an equal weight, that is the divided among the occurrences of this residue. So, if a residue is observed in a column that presents r distinct amino acids, and s occurrences of this amino acid, then it is assigned the weight $\frac{1}{rs}$. Then, for a sequence, the weights of its positions are summed to give the total sequence weight. Finally, weights are normalized to sum to 1.

Henikoff weights are rather easy to compute and provide satisfying results for pHMMs. This strategy corresponds to the default setting for internal weighting in HMMER and SAM, as stated in [Eddy and Wheeler, 2015] and [Hughey et al., 2003].

External weighting approaches Once the internal weights are set, the total weight assigned to the set of training sequences has to be adjusted. It can be seen as the number of independent observations that would be equivalent to the actual training set of sequences.

As for internal weighting, several external weighting strategies have been introduced. As described hereafter, we identified two main approaches: the number of effective sequences can be directly estimated from the set of sequences; or it can be adjusted to optimize the resulting protomaton.

Estimation of the number of effective sequences A first approximation of the number of independent sequences is based on the clustering method introduced earlier: sequences that are similar are clustered, and the final number of clusters is considered as the number of independent sequences. However, the number of clusters can vary depending on the choice of the representative of each cluster, and the computation might be costly.

Another simple estimation is given in [Altschul et al., 2009]. From an alignment of the sequences, the number N_I of independent sequences is estimated as the average number of distinct amino acids in the columns (gaps are treated as a 21st amino acid). One of the disadvantages of this method is that N_I can not get higher than 21.

To avoid this issue, another method was introduced in [Altschul et al., 2009]. Given a model where the amino acids in a columns have the probabilities $p_1, \ldots, p_{20}$, and are randomly and independently chosen, then the expected number of distinct amino acids N_A, in a column of size n is given by:

$$N_A = f(n) = 20 - \sum_{i=1}^{20} (1 - p_i)^n \tag{19}$$

The function f is monotonic and can be extended to any $n \in \mathbb{R}_+$, so it can be inverted, in order to estimate n from N_A, computed as the average over all the columns of the alignment of the sequences[1].

Target relative entropy Entropy can be defined as a measure of uncertainty. Let $p = (p_1, \ldots, p_N)$ be a distribution. Its entropy is mathematically defined with:

$$H(p) = -\sum_{i=1}^{N} p_i \log_2 p_i \tag{20}$$

The entropy of a model of a family of proteins can be defined as the average entropy of the distributions of the match states in the model. Then, the relative entropy of a model $\mathcal{M}$ compared to the random model $\mathcal{R}$ is defined as:

$$H(\mathcal{M}||\mathcal{R}) = \frac{1}{K} \sum_{i=1}^{K} \sum_{a=1}^{20} p_i(a) \log_2 \left(\frac{p_i(a)}{q_a} \right) \tag{21}$$

where K corresponds to the number of match states in the model. It can be shown that the relative entropy is always greater than or equal to 0 with equality if and only if $\mathcal{M} = \mathcal{R}$. It can be

[1]The function f depends on the choice of background probabilities, so no table is given for the estimation of n. However, the solution can easily be computed (with Newton's method for example)

interpreted as a distance between the models $\mathcal{M}$ and $\mathcal{R}$: the higher the relative entropy, the more specialized our model $\mathcal{M}$ [Durbin et al., 1998].

An external weighting strategy relies on the observation that the relative entropy of a model is related to the target distance between the model and the sequences of the family, that are wanted to be recognized ([Johnson, 2006]). Indeed, the more divergent the sequences, the more likely the mutations within the sequences, and thus the higher the uncertainty and the lower the relative entropy of the model. Moreover, if the sequences are attributed a higher total weight, the relative entropy will be increased, because the pseudo-counts will have less influence on the estimation of the emission probabilities. Therefore, [Johnson, 2006] and [Karplus et al., 1998] suggested to tune the total weight of the sequences in order to reach a target relative entropy of the model, corresponding to the distance of the scoring sequences. This target relative entropy (TRE) is usually empirically chosen ([Karplus et al., 1998], [Johnson, 2006]).

This external weighting strategy is the default set in both HMMER and SAM, that mostly aim at detecting remote homologs. For HMMER, an optimal target relative entropy was empirically determined. It is equal to 0.59 bits per column. As far as SAM is concerned, the average relative entropy per column, for an N column alignment is $50/\min(N, 140(1 - e^{-0.008N}))$ ([Karplus et al., 1998]).

As far as internal weighting is concerned, we decided to explore the tree-based approach that is performed by the program ClustalW, as it is the one currently used in Protomata. Besides, we made the choice of not looking any further in distance-based approaches, to the benefit of the position-based weighting suggested in [Henikoff and Henikoff, 1994], because it seems more consistent with our position-specific modeling strategy.

As for external weighting, we will investigate the number of effective sequences, thanks to the estimation performed with equation 19, and also the target relative entropy weighting, that is argued to enable the adjustment of the model depending on the distance of the sequences we want to recognize from the training set.

We have also considered other weighting strategies, that perform internal and external weighting all at once. These strategies aim at maximizing the entropy or the discrimination power of the computed model. They are described in appendix 6, in the case of pHMMs. We decided not to use them because they do not seem to adapt well to Protomata. Indeed, sequences are either assigned the maximum weight or a zero weight. So, as we are dealing with partial alignment, if a sequence weight is set to zero because its information is already given by other sequences and these other sequences are not aligned in the same PLAs, we will lack information in the estimation of the parameters ([Durbin et al., 1998] p 131).Moreover, some sequences might be treated as exceptions, independently of their weight. So, if a sequence is divergent and involved in very few PLAs, this weighting method will increase said sequence weight, without inducing any change in the protomaton.

3.2 Null-model selection

Given a set of training sequences, the computation of a protomaton returns the smoothed probability of each amino acid in every match state in PLAs, and the allowed transitions between blocs. Then, when a previously unseen sequence s is parsed, its probability given the model of the family

$\mathcal{M}$ is compared to its probability given a null model $\mathcal{R}$, leading to a log-likelihood ratio score S:

$$S(s) = \log \frac{\mathbb{P}(s|\mathcal{M})}{\mathbb{P}(s|\mathcal{R})} \tag{22}$$

So, if the sequence is more likely to have been generated by the null model, then its score is negative, otherwise it is positive. However, the null model has yet to be determined.

Background probabilities The first strategy, that is currently employed in Protomata, consists in using the background frequencies of the amino acids. Let us denote q_a the background frequency of amino acid a, for any sequence s:

$$\mathbb{P}(s|\mathcal{R}) = \prod_{a \in s} q_a \tag{23}$$

This random distribution is independent of the family being modeled and of the sequence being parsed.

With this null-model, the score of a sequence might be biased because of its composition. Indeed, if a family presents some rare amino acids, a sequence with these rare amino acids might get a rather high score, even though it does not belong to the family. Moreover, sequences with a different composition from the training sequences are very unlikely to be family members. Trained models should enable the rejection of sequences that do not belong to the family, but are very similar to its sequences. To that end, we explored alternative null-models, presented hereafter.

Family dependent null-models In order to reduce the composition bias, background probabilities can be adjusted to the distribution of amino acids in the training set of sequences. To go further, one can note that only conserved blocs are modeled, and the composition of the 'gap' positions, that are not characteristic of the family, might bias the previously describe null-model, especially in the case of long training sequences. A possibility is to consider the average distribution of amino acids over the match states of the computed protomaton , after regularization ([Hughey et al., 1997]).

Another approach, described in [Hughey et al., 1997], is argued to perform even better, at least as far as pHMMs are concerned. It consists in considering the geometric mean of the amino acid distributions in the match states, normalized to sum to unity. So, instead of considering a letter's average contribution to the probability, its average contribution to the sum of log-probabilities is actually taken into account. This null model should enable the discrimination of sequences from the family, that present characteristic sequences of amino acids, from non family members with the same amino acid composition.

Note that when one of the previous null-model is computed, it is also used as gap probabilities in the protomaton. Thus, the log-likelihood ratios in match states are changed, and the contribution of gaps to the final log-likelihood ratio score is still zero.

Family and sequence dependent null-model Although the geometric null model provides good results and was used as the default in SAM for several years, it appeared that it might not fully correct for compositional bias [Karplus et al., 2005a]. To tackle this issue, the idea of comparing the score of the parsed sequence with the score of the same sequence, but reversed, was introduced. The reversed sequence has several advantages over randomly generated sequences.

First, it has the same length and composition as the parsed sequence. Moreover, even though protein sequences are directed, reversed sequences are more likely to present features common in proteins, especially the undirected ones, than random sequences.

In this case, the probabilities of amino acids in gap states correspond to their background frequencies. Besides, note that as far as Protomata is concerned, there is no guarantee that the reversed-sequence will be aligned with the same path as the actual sequence.

3.3 Normalization of scores

As stated in subsection 2.2, depending on the path, a parsed sequence can be aligned with a various number of match states. In the case of a null-model at most family dependent (i.e. background frequencies, or average or geometric distribution of amino acids over match states), we have seen that gaps have a neutral contribution, and the score corresponds to the sum of the scores over match states in the alignment of the parsed sequence.

We study here the expectation of the log-likelihood ratio score of a sequence, for the null-models that are at most family-dependent (all but the reversed-sequence null-model).

Let us denote q_a the probability of a in the random distribution, and $p_i(a)$ its probability in a match state i. The expected score of the position of a random sequence aligned with match state i can be computed as follows:

$$\begin{aligned}
\mathbb{E}[X_i|\mathcal{R}] &= \sum_{a=1}^{20} \mathbb{P}(X = s_a) s_a \\
&= \sum_{a=1}^{20} q_a \log\left(\frac{p_i(a)}{q_a}\right) \\
&= -H(\mathcal{R}_i||\mathcal{M}_i) \\
&\leqslant 0
\end{aligned}$$

where $\mathcal{M}_i$ corresponds to the distribution of amino acids over the match state i. Therefore, the expectation of the score of a sequence, aligned with K match states $i_1, \ldots, i_K$ is given by:

$$\mathbb{E}[X|\mathcal{R}] = -\sum_{k=1}^{K} H(\mathcal{R}||\mathcal{M}_{i_k}) \tag{24}$$

Note that exceptions are possible in the case of Protomata, and there might be a path with no match state. The score of an alignment on this path is equal to zero. So, if a model presents a complete exception, all scores are positive, and therefore the expectation of scores of non-family members is positive.

Similarly, the expectation of the score of a sequence that belongs to the family, assuming that the alignment is always correct, is given by:

$$\mathbb{E}[X|\mathcal{M}] = \sum_{k=1}^{K} H(\mathcal{R}||\mathcal{M}_{i_k}) \geqslant 0 \tag{25}$$

Depending on the distributions in the protomaton, it seems that the more match states, the higher the absolute value of the score. This raises the issue of the comparability of scores: the

sequence of a family-member very well aligned with a small path might get a lower score than another sequence less well aligned with a longer path.

To tackle this issue, the score can be normalized by the number of match states encountered in the corresponding alignment. This way, instead of considering the total score of a sequence, we will deal with the average contribution of match states on the alignment, so the score should not depend on the length of the path anymore.

Note that the score of the most probable alignment is normalized, and it does not necessarily correspond to the alignment with the best normalized score.

As far as the reversed-sequence null-model is concerned, we have no guarantee that the reverse is aligned with the same path as the original sequence. So, both sequence and reverse might meet a different number of match states. However, if the reverse of a sequence is aligned with a complete exception, then the probability of the reverse corresponds to the product of the background probabilities of its amino acids (all aligned with a gap state), and the final score is equal to the log-likelihood ratio score computed with the background null model. So, if two sequences are aligned with two different paths, and their respective reverse are aligned with a complete exception, the same variability of scores occurs as with the background null-model, and there is a need for normalization. However, we did not find any immediate solution, and we will use the un-normalized score for this null-model.

4 Significance of scores for the classification of sequences

In the previous section, we described possible improvements in the computation of a protomaton and the scoring of new sequences. However, these modifications are useless if we can not classify sequences, and up to now, the choice of the treshold was left to the user. So, once the score of a sequence is computed, we seek its significance, defined hereafter, in order to predict whether or not the sequence belongs to the family.

First, we will give a clear definition of the significance of a score. Then, we will seek the distribution of scores, in order to compute the significance for background, family-dependent and reversed-sequence null-models.

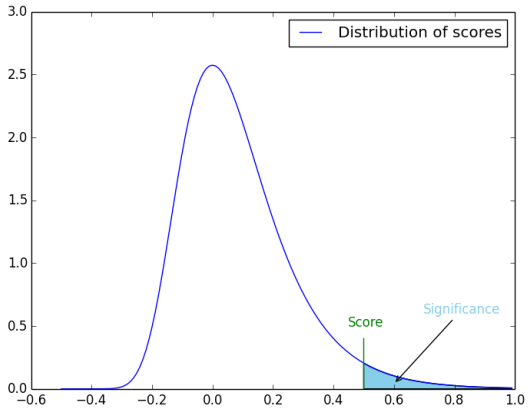
Hypothesis testing Given a protomaton computed from a set of sequences and a previously unseen sequence to parse, we are confronting two hypothesis:

$\mathcal{H}_0$: "The sequence comes from the null model,i.e. it does not belong to the family of proteins"

$\mathcal{H}_1$: "The sequence belongs to the family"

The level of significance α of the hypothesis test corresponds to the probability of wrongly rejecting $\mathcal{H}_0$. Let us denote S the random variable corresponding to the score of a sequence, under the hypothesis $\mathcal{H}_0$, and x the observed score for a parsed sequence. The significance α_x of the score x is given by:

$$\alpha_x = \mathbb{P}(S > x) \tag{26}$$



A significance threshold can then be set, for instance $\alpha = 0.05$, and a sequence will be accepted if and only if the significance of its score is lower than this threshold. In our example, it would mean that the risk of falsely accepting the sequence is lower than 5%. Of course, given the distribution of scores, the lowest score with significance lower than α can be computed. This value provides a threshold for new scores, in order to accept or reject previously unseen sequences.

Equation 26 suggests that we need to determine, or at least approximate, the distribution of scores (under the null-model hypothesis $\mathcal{H}_0$). This distribution clearly depends on the model and the scoring strategy used, but it scores of sequences of different lengths might also need to be considered separately. Indeed, the longer the sequence, the more likely we are to find an alignment that fits the model well, and hence the higher the score.

Given a protomaton, the Viterbi score is considered, so the score corresponds to the maximum of the scores on all possible paths, with exceptions counting as gaps. Let S be the random variable corresponding to the score of a sequence, and x a real number. Then:

$$\mathbb{P}(S > x) = \mathbb{P}(\max_{c \in \text{paths}} S_c > x) \quad \text{where } S_c \text{ denotes the Viterbi score on path } c \tag{27}$$

$$= 1 - \mathbb{P}(\max_{c \in \text{paths}} S_c \leqslant x) \tag{28}$$

$$= 1 - \prod_{c \in \text{paths}} \mathbb{P}(S_c \leqslant x) \quad \text{under the assumption that the paths are independent} \tag{29}$$

So, we reduced the search of the distribution on the protomaton to the search of the distribution on each path of the protomaton.[2]

Distribution of the score on a single path The results presented in this paragraph are valid for log-probability and log-likelihood ratio scores.

Let us consider a specific path c in a protomaton, and a null sequence for which we seek its score S_c on the path. For every $x \in \mathbb{R}$:

$$\mathbb{P}(S_c \leqslant x) = \mathbb{P}(\max_{l \text{ alignment}} S_c^l \leqslant x) \tag{30}$$

where S_c^l denotes the score of the alignment l of the sequence on path c. The random variables S_c^l are identically distributed (as shown below), and are assumed to be independent.

Given an alignment l, the score S_c^l can be expressed as:

$$S_c^l = X_1 + \cdots + X_N \tag{31}$$

[2]Note that the paths are not independent, especially because when building a protomaton, a choice can be made between a single path and a quite conserved bloc, and two paths with smaller (in the number of sequences involved) but highly conserved blocs. If the two blocs are chosen, the corresponding paths are very dependent. The need independence could then become a criterion in the setting of the Protomata.

where $\forall i \in [|1, N|], X_i$ denotes the random variable corresponding to the score in position i[3]. If a is an amino acid, $s_i(a)$ the score of a in position i and q_a the background probability of a[4], we have:

$$\mathbb{P}(X_i = s_i(a)) = q_a \tag{32}$$

which completely defines the distribution of the X_i.

Although the variables X_i are assumed to be independent, they are clearly not identically distributed, hence the usual central limit theorem can not be applied. However, it can be proved that the variables satisfy the hypotheses of the Lyapunov's central limit theorem. Hence, if μ_i and σ_i^2 respectively denote the expectation and variance of X_i, then:

$$\frac{1}{\sqrt{\sum_{i=1}^{n} \sigma_i^2}} \sum_{i=1}^{n} (X_i - \mu_i) \xrightarrow[n\to\infty]{d} \mathcal{N}(0,1) \tag{33}$$

where $\xrightarrow{d}$ denotes the convergence in distribution.

So, we will approximate S_c^l with a Gaussian distribution. One can note that this approximation might not perfectly fit the model, especially when there are only a few match states on the path.

Thus, he random variable $S_c = \max_{l \text{ alignment}} S_c^l$ corresponds to the maximum of L independent and identically distributed Gaussian distributions, where L denotes the number of possible alignments of the sequence on the path. Hence, S_c can be approximated with an extreme value distribution. The corresponding probability density function f and distribution function are, for $x \in \mathbb{R}$:

$$f(x) = \lambda \exp\left(-\lambda(x-\mu) - e^{-\lambda(x-\mu)}\right) \tag{34}$$

$$\mathbb{P}(S_c < x) = \exp\left(-e^{-\lambda(x-\mu)}\right) \tag{35}$$

where $\lambda > 0$ and μ are scale and location parameters.

According to [Eddy, 1997] and [Eddy, 2008], the parameters λ and μ depend both on the path and on the length of the sequence being parsed. However, there is no analytical way to determine the parameters. We implemented the maximum likelihood estimation described in [Eddy, 1997]. See Appendix 6 for more details.

Note that only the tail of the distribution is important here, so we can estimate the parameters so that the tail is fitted at best. Formulas are also given in Appendix 6.

Distribution of scores with protomata Given a protomaton and a sequence to parse, the distribution corresponding to the score is given by equation 29. This formula requires that we consider every single path of the protomaton, and estimate the parameters on each of these paths, which would be very costly, especially because there might be a lot of different paths and the computation on a single path is expensive, since a maximum likelihood estimation on random data is performed.

Moreover, a protomaton with several paths raises the question of the normalization of the score. Indeed, without normalization, a sequence that is very well aligned with a few characteristic

[3]Note that we are restricting ourselves to path c, for which we know the number of match states, so the different scores can be normalized by N or not

[4]We are seeking the distribution of the scores for all the proteins

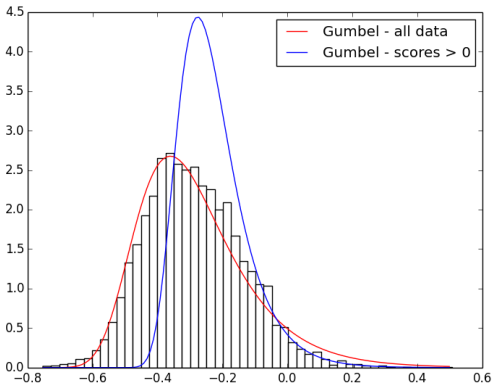


Figure 7: Distribution of scores of 5000 random sequences of length 300 on a protomaton modeling the family of ORs, compared to the Gumbel distributions estimated on the whole histogram, and on the positive scores only.

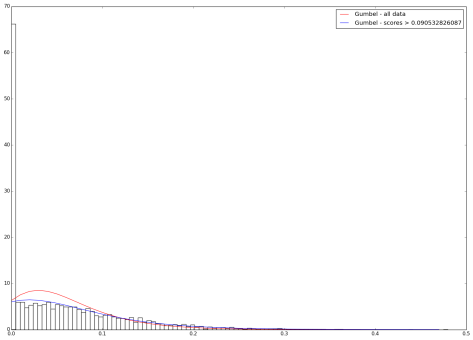


Figure 8: Distribution of scores of 5000 random sequences of length 900 on a protomaton modeling the family of TNFs, compared to the Gumbel distributions estimated on the whole histogram, and on the histogram censored up to the third quartile

match states might get rejected, because its unnormalized score is lower than the score of random sequences that align less well with a less characteristic but longer path. Therefore, we will now only consider normalized scores (by the number of match states), so the score corresponds the average contribution per match states on the best alignment path. This way, the scores will be more similar and their distribution will not be biased by the variability of path lengths.

Thanks to the normalization, and in order to avoid expensive computation, let us consider the score of a parsed sequence. Let us remind that we are considering a log-probabilities or a log-likelihood ratio score. This score corresponds to a maximum of (assumed to be) independent Gaussian distributions, that are not identically distributed. However, it seems reasonable to assume that match columns present expectation and variance of scores on different paths that are not significantly different, and the distribution of scores might still be accurately estimated by a Gumbel distribution, with parameters to be estimated as before.

Log-likelihood ratio scores The previous discussion is valid for log-likelihood ratio scores. Specifically, it can be applied to scores computed with the background probabilities null-model, but also the family dependent null-models, described in the previous section (average and geometric distributions of amino acids over match states).

So, we will try to approximate the distribution of scores computed in this context with a Gumbel distribution. However, because of all the approximations made before, the estimation of the whole distribution of scores is not as precise as it is for scores on a single path, as with pHMMs [Eddy, 2008]. Moreover, as exceptions are treated as gaps, scores on a protomaton tend to be higher, or even exclusively positive. This case occurs when a whole training sequence is treated as an exception for example.

Besides, as far as the significance is concerned, only the tail of the distribution is of interest. So, we will fit the distribution only on the highest scores. The proportion of scores considered can be set, and has to be higher than the risk α that is set to accept of reject a parsed sequence.

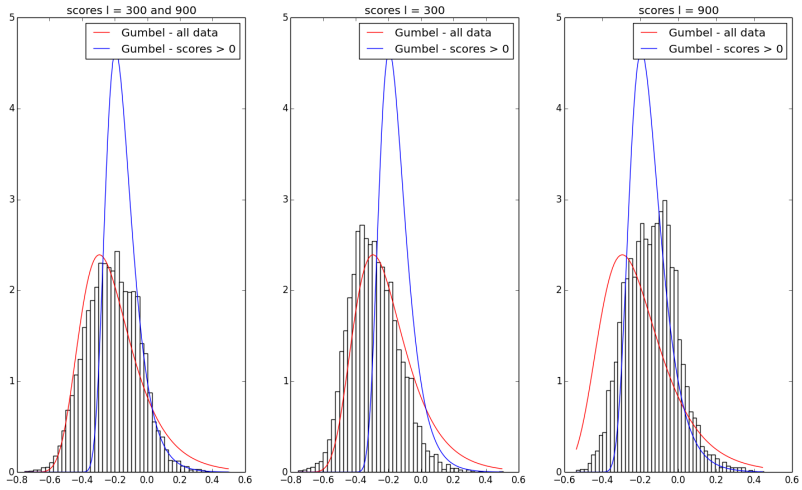


Figure 9: Scores distributions for 10000 sequences of lengths 300 and 900, for 5000 sequences of length 300 and 5000 sequences of lengths 900, compared with the Gumbel distributions estimated on scores for sequences of lengths 300 and 900 on a protomaton modeling the family of ORs.

We compare histograms of scores and the corresponding estimated distributions, to evaluate the accuracy of the approximation by a Gumbel distribution. To do so, we used two data sets, from the family of olfactory receptors (OR) and tumor necrosis factors (TNF). These sets are described in section 5.

As we can see on figure 7, the censored estimation of the tail of the distribution with a Gumbel distribution provides an accurate estimation of the distribution of the highest scores. The need of an estimation of the tail is even more important when the protomaton presents a complete exception, as shown in figure 8, where all sequences have a positive score.

The distribution does depend on the length of the sequence, as shown in figure 9: a longer sequence is more likely to get a higher score. However, if the sequences are almost of the same length, then the Gumbel distribution might fit the scores well enough. We will estimate the distribution of scores assuming that the parsed sequences will be of approximately the same length as the training sequences.

Plots presented in this section were made with scores computed with the background null-models. The same comparison of scores to the corresponding estimated Gumbel distribution was made for the other family-dependent null-models and it lead to similar plots.

Reversed-sequence null model We have determined the distribution of scores for background and family dependent null-models. However, as far as the reversed-sequence null-model is concerned, a new strategy is required [Karplus et al., 2005b].

Results of the previous paragraphs also apply to log-scores: log-scores of random sequences follow a Gumbel distribution of parameters λ and μ. So, the reversed sequence being assumed to also be a protein, with same length and composition of the parsed sequence, its score follows the same distribution.

Let us consider a sequence, and let us denote s_M the log-score of said sequence aligned with the model, and s'_M the score of the corresponding reversed sequence. We have, if f denotes the probability density function of scores, for all $x \in \mathbb{R}$:

$$f_{s_M}(x) = f_{s'_M} = \lambda \exp\left(-\lambda(x-\mu) - e^{-\lambda(x-\mu)}\right) \tag{36}$$

Let $S = s_M - s'_M$ be the final score of the sequence (with the reversed-sequence null-model). Assuming the random variables s_M and $-s'_M$ are independent, the density function of the sum of these variables f_S can be computed as the convolution product of their respective densities f_{s_M} and $f_{-s'_M}$, leading to:

$$f_S(a) = \frac{-\lambda e^{-\lambda a}}{\left(1+e^{-\lambda a}\right)^2} \tag{37}$$

Then the probability mass function can be deduced from the density, for $x \in \mathbb{R}$:

$$\mathbb{P}(S < x) = \int_{-\infty}^{x} f_S(t)dt = \int_{-\infty}^{x} \frac{-\lambda e^{-\lambda t}}{\left(1+e^{-\lambda t}\right)^2} dt \tag{38}$$

$$= \frac{1}{1+e^{-\lambda x}} \tag{39}$$

This corresponds to a sigmoid distribution. The λ parameter can be calibrated thanks to a maximum-likelihood estimation as before [Karplus et al., 2005b], from a large sample of scores. The λ parameter should not depend on the length of the model or the sequence, so it can be computed once for the model, and the used for any significance computation given a new sequence. The sigmoid distribution seem to provide an accurate estimation of the distribution of scores of a protomaton, as shown in figure 10 for the family of ORs, even though the protomaton presents different paths of various lengths.

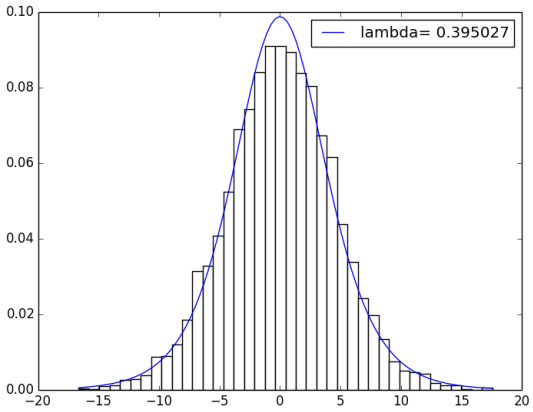


Figure 10: Histogram of the scores of 5000 random sequences of length 300 computed with a protomaton representing the family of ORs, with the reversed-sequence null-model. The distribution is compared to the corresponding estimated sigmoid distribution.

In the case of the family of TNFs, as the protomaton presents a complete exception, random sequences might often be aligned with the exception, leading to a zero score, as shown in figure 11. So, there is a discontinuity in the sigmoid distribution. However, the sigmoid still provides an accurate estimation of the distribution of non-zero scores, as we can see in figure 12. In that case, we can either consider non-zero scores and adjust the risk threshold, or estimate the probability p_0 of the score being equal to zero, and use the law of total probabilities:

$$\mathbb{P}(S > x) = \mathbb{P}(S > x | S = 0)p_0 + \mathbb{P}(S > x | S \neq 0)(1 - p_0) \tag{40}$$

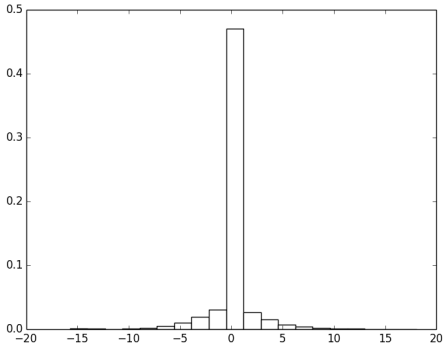


Figure 11: Histogram representing the distribution of the scores of 5000 sequences of length 300 computed on a protomaton representing the family of TNFs, using the reversed-sequence null-model. The protomaton presents a complete exception, which explains the great number of zero scores.

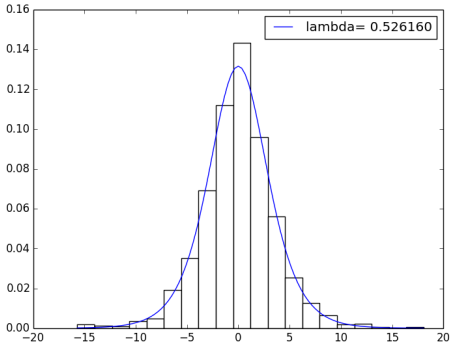


Figure 12: Histogram of non-zero scores of random sequences of length 300, computed on a protomaton representing the family of TNFs, using the reversed sequence null model. The normed histogram is compared to the estimated sigmoid distribution.

In the case of a complete exception, the λ parameters of the sigmoid distribution does not depend on the length of the parsed sequences. Indeed, the longer the sequence, the longer its reverse, so the length bias should be compensated with this null-model. However, when there is a complete exception, the probability of a zero score does depend on the length of the sequence. As before, we will estimate the distribution parameters on a sample of sequences of the same length as the training set.

5 Experiments

In this section, we present preliminary experiments to compare the different null-models and weighting strategies. Further experiments on other families are still running, but results were not available.

5.1 Implementation

To perform these experiments, I used the program `Protomata-Learner`, available on the cluster of the Genouest bioinformatics platform and accessible through a web interface[5]. To augment `Protomata-learner` with the new features introduced before, I have implemented the new weighting schemes, scoring strategies and significance estimation as three additional modules in Python. The first module is for weighting the training sequences. I implemented the position-based internal weighting strategy. Moreover, it includes the computation of the number of effective sequences, based on equation 19. As for the target relative entropy weighting strategy, it seemed reasonable for the external weight to be between one sequence and the actual number of training sequences. This enables a dichotomic search for the external weight that reaches the target relative entropy set by the user, with a precision that can also be specified.

[5] `http://tools.genouest.org/tools/protomata/learn/`

The second module computes the score for the reversed-sequence null-model, and the normalized scores for the other null-models. This required the computation of the average distribution and average geometric distribution of amino acids over match states of the protomaton.
The third module deals with the estimation of the distribution of scores. It performs the generation of a sample of scores of random sequences. Current implementation generates these sequences using a set of proteins from the Uniref database[6]. To generate a random sequence of length l, a sequence is taken at random in the data set, truncated and shuffled. The random sequences are then scored and I implemented the maximum likelihood estimations of the distribution parameters, for background, family-dependent and reversed-sequence null-models. Then I added the computation of the significance of scores in the scoring program.

5.2 Description of data sets

In order to validate the scoring strategy and compare the different weighting methods and null-models, we used two different data sets, available at:
`http://www.irisa.fr/dyliss/benchmark-protomata-scores`.

A first set is composed of sequences of proteins from the family of the tumor necrosis factors (TNF). These proteins are of interest because of their implication in cancer pathologies. The set at hand consists in 18 family members, described in the file `tnfsf_human_pos.fasta`, and 20 counter-examples, contained in `tnfsf_neg.fasta`. We used the set of parameters described in table 1 for `Protomata-learner`. These parameters were established by an expert as appearing to provide an accurate and robust alignment. The default is used for the other parameters.

parameter	value
Significance threshold	2
Consensus	weak
Percentage quorum	18
Dirichlet mixture	byst-4.5-0-3.9comp

Table 1: Set of parameters used with `Protomata-Learner` for the family of TNFs.

This set of sequences is known to perform well with Protomata, and to lead to rather small and reliable automata.

The other data set consists in sequences from the family of olfactory receptors (OR). We used 74 sequences, contained in `RO_pos.fasta` and 34 counter-examples, contained in the file `RO_neg.fasta`. Again, a set of parameters that provide reliable and robust alignments was provided. It is described in table 2. This family is more complex to model than the previous one, especially because it presents more discrepancies, and the resulting automata presents a lot of different paths, with various numbers of

parameter	value
Significance threshold	3
Max. bloc size	25
Consensus	weak
Percentage quorum	10
Dirichlet mixture	recode3.20comp

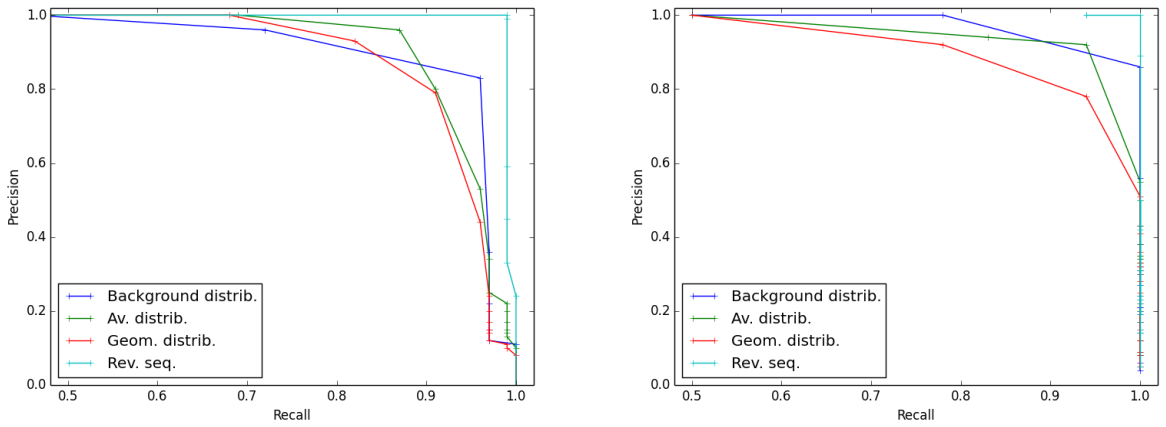
Table 2: Set of parameters used with `Protomata-Learner` for the family of ORs.

[6] I acquired this data set from the Uniref database, on the 18th of April 2017, with the following request: `'length:[50 TO 5000] NOT b NOT z NOT 10 reviewed:yes AND identity:0.5'`. A file containing the sequence identifiers is available at `http://www.irisa.fr/dyliss/benchmark-protomata-scores`

match states.

To confront different models and scoring strategies, we will compute the precision and recall values for different thresholds of significance, and compare the resulting precision-recall curves. We can also compare the areas under these curves: the greater the area, the better the model and/or scoring strategy.

5.3 Comparison of the different null-models



(a) Family of olfactory receptors

(b) Family of tumor necrosis factors

Figure 13: Precision/recall curves for the classification of sequences from the families of ORs and TNFs, for different null-models (background probabilities, average distribution of amino-acids over match states, average geometric distribution over match states, and reversed-sequence null-model)

To compare the efficiency of the scoring strategies, and significance estimation provided by the different null-models, we use the position-based internal weighting and the number of effective sequences to weight the training sequences, as it seems to be the most neutral weighting strategy.

As far as the set of TNFs is concerned, we performed a leave-one-out cross validation, as few sequences are available and are already sharing a reasonable sequence identity. As for the family of ORs, we also aim at using a leave-one-out cross validation. However, there are more sequences, and some of them are redundant. So, for each removed sequence, there is a possibility that very similar sequences remain in the training set, which would bias the results. To address this issue, we filtered the data set so that the remaining sequences share less than 30% of similarities, and performed the leave-one-out cross validation on this reduced set, named `RO_pos_nr70.fasta`.

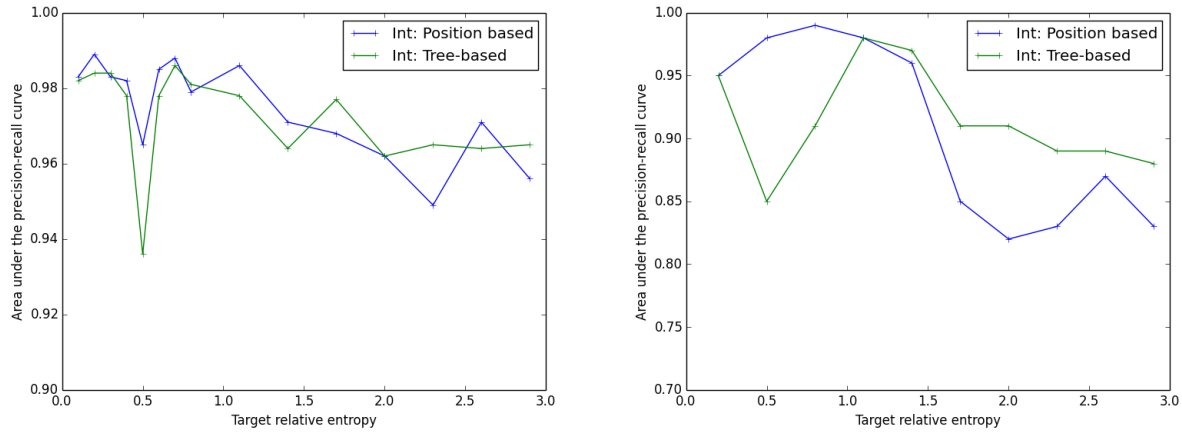
The results of the cross validation are displayed in figure 13. It seems that the family dependent null-models perform better than the background null-model. The average distribution and average geometric distribution of amino acids over match states seem to lead to similar outcomes, and they are both outperformed by the reversed-sequence null model, that seems to provide a very efficient prediction.

We can try to explain why the family dependent null-models do not produce as good outcomes as the reversed-sequence null-model. First, the amino acid distributions are estimated on the match

states of the protomaton, and used for both the random distribution and gap probabilities. However, there might be only a few match states that are not representative of the composition in amino acids of the family sequences. Moreover, we have seen that with these models, the distribution of scores depends on the length of the sequences, and we only consider the distribution of scores for sequences of the same length as the training sequences. More precise distributions of scores can be approximated for several lengths and used to compute the significance, but testing this strategy would be computationally expensive.

Note that the reversed-sequence null model might also be improved by forcing the reversed-sequence to be aligned to the same path as the sequence, but this kind of parsing is not implemented.

5.4 Comparison of the different weighting strategies



(a) Family of olfactory receptors

(b) Family of tumor necrosis factors

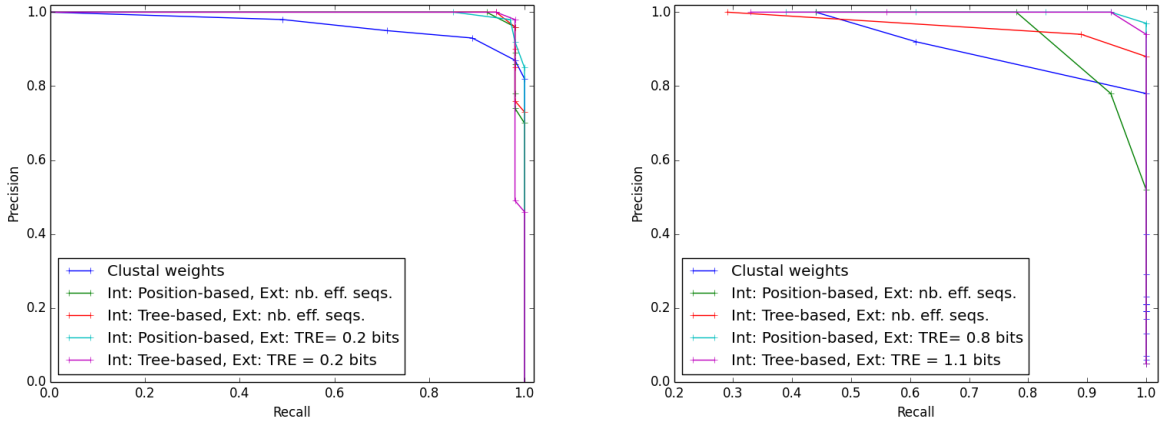
Figure 14: Area under the precision-recall curve for different relative entropy targets for the families of ORs and TNFs, for both position-based and tree-based internal weighting.

To compare the different weighting strategies, we used a leave-one-out cross validation for the family of TNFs. As before, for the family of ORs, there is a risk of having a training sequence very similar to a test sequence, and therefore biasing the results. However, we want to test our sequence weighting strategies, and assess their performances when dealing with similar sequences in the training set. So, instead of filtering the data so that it is non-redundant as in section 5.3, we used a three-fold cross validation, and for each fold, in each training set, we removed the sequences that were more than 30% similar to the sequences in the test set. Moreover, we filtered the test sets (positive and negative) so that they are non-redundant at 70%. For each fold step $k \in \{1, 2, 3\}$, we used the training set `RO_pos_train_foldk.fasta`, and the test sets `RO_pos_test_foldk_nr70.fasta` and `RO_neg_nr70.fasta`.

We used the normalized scores computed with the background null-model.

First, we considered the target relative entropy weighting scheme. We compare the model efficiencies for different targets, and tree-based and position-based internal weighting strategies. As

shown in figure 14, it seems that the best TRE for the family of TNFs is 0.2 with the position-based internal weighting and 0.8 with the tree-based internal weighting. As for the family of ORs, it appears the the best results are reached for a target of 0.2 for both internal weighting approaches.



(a) Family of olfactory receptors

(b) Family of tumor necrosis factors

Figure 15: Precision-recall curve for the different internal and external weighting strategies applied to the families of ORs and TNFs.

Then, we compared these TRE external weighting with the weighting scheme currently used in Protomata, that consists in normalizing the weights from `ClustalW` so that the highest score is equal to 1, and the number of effective sequences. As displayed in figure 15, it appears that the former weighting strategy used in Protomata leads to the worst results. Moreover, the target relative entropy weighting strategy provides the best results. Note that these outcomes required the search for an additional parameter.

So, we compared the previously proposed weighting strategies and null-models on two different data sets. Obtained results seem to indicate that the strategies that are currently used in Protomata are outperformed by all the suggestions made here. More precisely, position-based or tree-based internal weighting strategies seem to give quite similar outcomes. However, the external weighting scheme can be improved thanks to the number of effective sequences or the target relative entropy weighting scheme. This latter strategy, however, requires the setting of one more parameter.

Other data sets of proteins are being considered, and more specifically families of enzymes, that perform the catalysis of chemical reactions. Validation tests will also be run on these families, to confirm (or not) previous results.

6 Conclusion

Sequencing provides an exponentially growing database of protein sequences, gathered in families. The task at hand is to design predictive models for these families, that can be used to recognize new members.

The current state-of-the-art model on proteins is the pHMM. It describes conserved positions within the family and allows insertions and deletions. The emission probabilities embed background knowledge on amino acid distributions, using Dirichlet mixtures, leading to very accurate models, even when few training sequences are available.

More recently, the Dyliss team introduced a new approach, called Protomata. They propose to design an automaton modeling conserved regions, and indicating the allowed successions of these conservations. This model is more expressive than pHMMs, because it enables the representation of possible discrepancies between the sequences of the family. Protomata weights were computed with the same techniques as pHMMs.

Despite good results, the multiplicity of paths offered by Protomata leads to a great variability of the scores of parsed sequences, making it difficult to compare scores and set a threshold for classification. Besides, the current sequence weighting strategy used in Protomata is rather basic, and the only null-model available corresponds to the background distribution of amino acids.

During my internship, I investigated sequence weighting and null-model approaches developed for pHMMs, and suggested to adapt several internal and external weighting schemes to Protomata, along with more accurate null-models, that depend on the family, or even on the sequence being parsed. Moreover, I suggested a normalization of the score by the number of match states, to reduce variability and I proposed a method to estimate the significance of these scores, from a sample of random sequences.

Thanks to the computation of the significance, I was able to assess the prediction power of models thanks to prediction-recall curves. I used this discrimination power to compare the different sequence weighting strategies and null-models. It seems that tree-based and position-based internal weighting approaches lead to similar results, and the target relative entropy external weighting enables the computation of more accurate models. Note that this external weighting scheme requires the user to choose this target. Further work could focus on the estimation of this parameter from the set of training sequences. Furthermore, the reversed-sequence null-model appears to perform better than family-dependent null-models. However, the distributions of scores for these random models depend on the length of the sequences and better results might be achieved if the length of the sequence was taken into account in the estimation of the significance.

Although the presented experiments seem to indicate an enhancement of the Protomata models, further tests will be needed to validate these results. Moreover, I only compared the different strategies to each other, and it might be interesting to see what combinations of parameters lead to the best prediction powers.

References

[Altschul, 1991] Altschul, S. (1991). Amino acid substitution matrices for an information theoretic perspective. *J. Mol. Biol*, 259:555–565.

[Altschul et al., 2009] Altschul, S., Gertz, E., Agarwala, R., Schffer, A., and Yu, Y.-K. (2009). PSI-BLAST pseudocounts and the minimum description length principle. *Nucleic Acid Research*, 37(3):815–824.

[Altschul et al., 2010] Altschul, S., Wootton, J., Zaslavsky, E., and Yu, Y.-K. (2010). The construction and use of log-odds substitution scores for multiple sequence alignment. *PLoS Comput. Biol.*, 6(7):1–17.

[Bailey and Gribskov, 1997] Bailey, T. L. and Gribskov, M. (1997). Score distributions for simultaneous matching to multiple motifs. *Journal of computational biology*, 4(1):45–59.

[Brown et al., 1993] Brown, M., Hughey, R., Krogh, A., Mian, I., Sjlander, K., and Haussler, D. (1993). Using Dirichlet mixture priors to derive hidden Markov models for protein families. *Proc. First Int. Conf. Intell. Sys. Mol. Biol.*, pages 47–55.

[Coste, 2016] Coste, F. (2016). Learning the language of biological sequences. In Heinz, J. and Sempere., J., editors, *Molecular biology of bacteriophage T4*. Springer-Verlag.

[Coste and Kerbellec, 2006] Coste, F. and Kerbellec, G. (2006). Learning automata on protein sequences. *JOBIM*, pages 199–210.

[Durbin et al., 1998] Durbin, R., Eddy, S., Krogh, A., and Mitchison, G. (1998). *Biological sequence analysis*. Cambridge University Press.

[Eddy et al., 1995] Eddy, S., Mitchison, G., and Durbin, R. (1995). Maximum discrimination hidden Markov models of sequence consensus. *J. Comput. Biol.*

[Eddy and Wheeler, 2015] Eddy, S. and Wheeler, T. J. (2015). *HMMER User's Guide*. Howard Hughes Medical Institute.

[Eddy, 1997] Eddy, S. R. (1997). Maximum likelihood fitting of extreme value distributions.

[Eddy, 2008] Eddy, S. R. (2008). A probabilistic model of local sequence alignment that simplifies statistical significance estimation. *PLOS Computational Biology*, 4(5):1–14.

[Gerstein et al., 1994] Gerstein, M., Sonnhammer, E., and Chothia, C. (1994). Volume changes in protein evolution. *J. Mol. Biol*, 236:1067–1078.

[Grundy et al., 1997] Grundy, W., Bailey, T., Elken, C., and Baker, M. (1997). Meta-MEME: Motif-based hidden Markov models of protein families. *Comput. Appl. Biosci.*, 13(4):397–406.

[Henikoff and Henikoff, 1992] Henikoff, S. and Henikoff, J. (1992). Amino acid substitution matrices from protein blocks. *Proc. Natl. Acad. Sci.*, 89:10915–10919.

[Henikoff and Henikoff, 1994] Henikoff, S. and Henikoff, J. (1994). Position based sequence weights. *J. Mol. Biol*, 243:574–578.

[Henikoff and Henikoff, 1996] Henikoff, S. and Henikoff, J. (1996). Using substitution probabilities to improve position specific scoring matrices. *Computer applications in the Biosciences*, 12(2):135–143.

[Hughey et al., 1997] Hughey, R., Barrett, C., and Karplus, K. (1997). Scoring hidden Markov models. *Comput. Appl. Biosci.*, 13(2):191–199.

[Hughey et al., 2003] Hughey, R., Karplus, K., and Krogh, A. (2003). *SAM: sequence alignment and modeling software system*. Baskin Center for Computer Engineering and Science.

[Johnson, 2006] Johnson, S. (2006). *Remote protein homology detection using hidden Markov models*. PhD thesis, Washington university.

[Karlin and Altschul, 1990] Karlin, S. and Altschul, S. F. (1990). Methods for assessing the statistical significance of molecular sequence features by using general scoring schemes. *Proc. Natl. Acad. Sci.*, 87:2264–2268.

[Karplus et al., 1998] Karplus, K., Barrett, C., and Hughey, R. (1998). Hidden markov models for detecting remote protein homologies. *Bioinformatics*, 14(10):846–856.

[Karplus et al., 2005a] Karplus, K., Karchin, R., Shackelford, G., and Hughey, R. (2005a). Calibrating E-values for hidden Markov models using reverse-sequence null models. *Bioinformatics*, 21(22):4107–4115.

[Karplus et al., 2005b] Karplus, K., Karchin, R., shackelford, G., and Hughey, R. (2005b). Calibrating E-values for hidden Markov models using reverse-sequence null models. *Bioinformatics*, 21(22):4107–4115.

[Kerbellec, 2008] Kerbellec, G. (2008). *Apprentissage d'automates modelisant des familles de sequences proteiques*. PhD thesis, Universite de Rennes 1.

[Krogh and Mitchison, 1995] Krogh, A. and Mitchison, G. (1995). Maximum entropy weighting of aligned sequences of proteins or dna. pages 215–221.

[Nguyen et al., 2013] Nguyen, V.-A., Boyd-Graber, J., and Altschul, S. (2013). Dirichlet mixtures, the Dirichlet process, and the structure of protein space. *Journal of computational biology*, 20(1):1–18.

[Rabiner, 1989] Rabiner, L. R. (1989). A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286.

[Sjolander et al., 1996] Sjolander, K., Karplus, K., Brown, M., Hughey, R., Krogh, A., Mian, I., and Haussler, D. (1996). Dirichlet mixtures : a method for improved detection of weak but significant protein sequence homology. *Comput. Appl. Biosci.*, 12:327–345.

[Sunyaev et al., 1999] Sunyaev, S., Eisenhaber, F., Rodchenkov, I., Eisenhaber, B., and ans E.N. Kuznetsov, V. T. (1999). PSIC: profile extraction from sequence alignments with position-specific counts of independant observations. *Protein Engineering*, 12(5):387–394.

[Thompson et al., 1994] Thompson, J., Higgins, D., and Gibson, T. (1994). Clustal W: Improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position specific gap penalties and weight matrix choice. *Nucleic Acids Research*, 22(22):4673–4680.

[Ye et al., 2011] Ye, X., Yu, Y.-K., and Altschul, S. (2011). On the inference of Dirichlet mixture priors for protein sequence comparison. *Journal of computational biology*, 18(8):941–954.

Appendix A Inference of Dirichlet priors

Dirichlet priors are used to embed background knowledge in the estimation of the emission probabilities in Protomata.

A lot of sequence alignments are available. A data set is created with the columns of these alignments, that will be used to infer prior knowledge about proteins and amino acids. The Dirichlet mixture is chosen as the mixture that maximizes the probability of the columns. There is no algorithm to compute the exact solution, so several approximation approaches were introduced.
The first estimation was presented in [Sjolander et al., 1996]. The number M of components is fixed. They seek the mixture with parameters Θ that maximizes the likelihood function $\mathcal{L}(\Theta) = \prod_{i=1}^{N} \mathbb{P}(\vec{o}_i|\Theta)$, and hence minimizes the encoding cost:

$$DL(D|\Theta) = -\sum_{i=1}^{N} \log \mathbb{P}(\vec{o}_i|\Theta) \tag{41}$$

where N denotes the number of columns in the data set. For the optimization, a gradient-based descent is used.
This method provides an estimate of a local optimum, however, there is no guarantee that the optimum found is global.

Another approach was described in [Ye et al., 2011], where the number of components is also fixed. The optimization is performed using a Gibbs-sampling algorithm, and a minimization of the description length of the data, especially for the setting of the concentration parameters $|\vec{\alpha}_j|$:

$$DL(D|\Theta) = -\sum_{i=1}^{N} \log \sum_{j=1}^{M} q_j \mathbb{P}(\vec{o}_i|\vec{\alpha}_j) \tag{42}$$

These approaches differ in two ways. The first difference concerns the likelihood functions to be optimized. Their expressions are slightly different, but the main divergence comes from the fact that it is either calculated for the particular observations in the columns ([Ye et al., 2011]) of for the count vectors in the columns ([Sjolander et al., 1996]). However, the difference produced for any column of observation depends only upon the column's count vector, and not upon the parameters that are optimized, so the result should not be affected. The second divergence comes from the choice of the number of components. It is indeed important as too many components might lead to over-fitting, but not enough might not provide a good description of the columns. In [Sjolander et al., 1996], mixtures were computed with any number of components between 1 and 30, and were used for experimentations, and the mixture that gives the most satisfying results is claimed to be the best one. In [Ye et al., 2011], the MDL was used. An estimate of the complexity of a mixture model is seeked, and the best Dirichlet mixture is the one that minimizes the sum of the complexity of the model and the description length of the data given the model.

The most recent inference approach is presented in [Nguyen et al., 2013]. The main difference from previous methods comes from the fact that the number of components is not fixed, and the optimization is performed thanks to the Dirichlet process. As before, the concentration parameters are set to minimize the description length of the data (equation 42). The resulting Dirichlet mixtures are very different from the previous ones, as the mixture that is claimed to provide the best

accuracy has about 600 components (the previous methods lead to mixtures with no more than 30 components). As 600 components might lead to longer computations, a compromise between computation efficiency and accuracy is provided with a mixture of 134 components.

Dirichlet mixtures correspond to the state-of-the-art method for the elaboration of pseudo-counts in the estimation of emission probabilities. There is no algorithm to compute the best mixture from a set of columns, but several approximation techniques are available. This leads to several mixtures, even for a fixed number of components that all provide satisfying results. Although the 134-component mixture from [Nguyen et al., 2013] should provide a better model of the alignment columns of proteins, it is still not used in the statistical modeling of families of proteins, in favor of mixtures with less components[7].

Appendix B Maximum entropy and maximum discrimination sequence weighting

The sequence weighting strategies presented in subsection 3.1 involve separate internal and external weighting. Other weighting approaches are possible: the following weighting approaches ([Krogh and Mitchison, 1995]) perform the estimation of the relative and absolute weights of the sequences at the same time. The weights will not only depend on the set of sequences, but they will be set in order to optimize the model of the family.

A first approach is based on the maximization of the entropy of the model, introduced earlier:

$$H = -\frac{1}{K}\sum_{i=1}^{K}\sum_{a=1}^{20} p_i(a)\log_2(p_i(a)) \tag{43}$$

where K corresponds to the number of columns. The entropy depends on the estimation of the emission probabilities, that depend on the weights. [Krogh and Mitchison, 1995] suggested to weight sequences to maximize the entropy of the model, in order to get a model as general as possible, with the maximum entropy weights ω^{ME}:

$$\omega^{ME} = \arg\max_{\omega} H(\omega) \tag{44}$$

To do so, a gradient descent approach is used, that requires to calculation of the derivative of the entropy function, which itself requires the calculation of the derivative of the emission probabilities:

$$p_i(a) = \sum_{j=1}^{M} \mathbb{P}(\vec{\alpha}_j|\vec{o}_i,\Theta)\frac{\sum_n \omega_n(o_i^n(a)+\frac{1}{N}\alpha_j(a))}{\sum_{a'}\sum_n \omega_n(o_i^n(a')+\frac{1}{N}\alpha_i(a'))} \tag{45}$$

where $o_i^n(a) = 1$ when the observed amino acid of the sequence n in position i is a, and $= 0$ otherwise, N denotes the number of training sequences, ω_n the weight assigned to the n-th sequence. Moreover, we have:

$$\mathbb{P}(\vec{\alpha}_j|o_i(a),\Theta) = \frac{m_j\mathbb{P}(\vec{o}_i|\vec{\alpha_j})}{\sum_{k=1}^{M} m_k\mathbb{P}(\vec{o}_i|\vec{\alpha_k})} \tag{46}$$

[7]The mixtures that are used for statistical modeling of families of proteins are available at https://compbio.soe.ucsc.edu/dirichlets/ , the number of components never exceeds 20.

$$\mathbb{P}(\vec{o_i}|\vec{\alpha_k}) = \frac{\Gamma(\vec{\alpha_j})}{\Gamma(\sum_a \sum_n (\omega_n o_i^n(a)) + |\vec{\alpha}_k|)} \prod_{a=1}^{20} \frac{\Gamma(\sum_a (\sum_n (\omega_n o_i^n(a)) + \alpha_k(a))}{\Gamma(\alpha_k(a))} \tag{47}$$

With the derivative, the weights can be adjusted as follows:

$$\omega_n^{new} \leftarrow \omega_n^{old} + \epsilon \frac{\partial H}{\partial \omega_n} \tag{48}$$

Through the maximization of entropy, this method tends to give higher weights to divergent sequences, and therefore it sets the internal weights.

Moreover, it is proven that sequences are either assigned a zero weight, or they have non-zero weights and the same probability in the model. So, the method can be used to remove close homologs from the set of training sequences. However, even if sequences are similar, they are still important and this weighting scheme can lead to a non-negligible loss if information.

Another approach is based on the discrimination power of the model. As stated in [Eddy et al., 1995], given a set S of training sequences, the maximum a posteriori (MAP) solution is seeked, i.e.:

$$\mathcal{M} = \arg\max_{\mathcal{M}'} \mathbb{P}(\mathcal{M}'|S) \tag{49}$$

The application of Bayes rule leads to:

$$\mathcal{M} = \arg\max_{\mathcal{M}'} \frac{\mathbb{P}(S|\mathcal{M}')\mathbb{P}(\mathcal{M}')}{\mathbb{P}(S)} \tag{50}$$

As $\mathbb{P}(S)$ is rather difficult to calculate, the solution is also often approximated through maximum likelihood.

For maximum discrimination, sequences are supposed to belong either to the model $\mathcal{M}$ or to the null model $\mathcal{R}$. The MAP gives:

$$\mathcal{M} = \arg\max_{\mathcal{M}'} \prod_{n=1}^{N} \mathbb{P}(\mathcal{M}'|S_n) = \arg\max_{\mathcal{M}'} \prod_{n=1}^{N} \frac{\mathbb{P}(S_n|\mathcal{M}')\mathbb{P}(\mathcal{M}')}{\mathbb{P}(S_n|\mathcal{M}')\mathbb{P}(\mathcal{M}') + \mathbb{P}(S_n|\mathcal{R})\mathbb{P}(\mathcal{R})} \tag{51}$$

Assuming all models are equiprobable, the we aim at maximizing the following function, called discrimination power of the model in [Eddy et al., 1995]:

$$D(\mathcal{M}) = -\sum_n \lg \frac{\mathbb{P}(S_n|\mathcal{M})}{\mathbb{P}(S_n|\mathcal{M}) + \mathbb{P}(S_n|R)} \tag{52}$$

where $S_1, \ldots, S_N$ denotes the set of training sequences and R denotes the random model.

As before, weights are set through a gradient descent. Poorly represented sequences are attributed a higher weight, so that their score in the model will be sufficiently high compared to random.

Both method are said to lead to comparable results on pHMMs. They are described in the case where the emission probabilities are estimated through maximum likelihood, so only relative weights matter.

Position-specific counts of independent observations Note that there exists also a method proposed by [Sunyaev et al., 1999], that assigns a position-specific weight to each sequence position, arguing that the position weight should be dependent on the environment of the corresponding residue, in the final 3D structure of the protein.

Appendix C Lyapunov's central limit theorem

Theorem 1. *Lyapunov's central limit theorem Let $(X_1, X_2, \dots)$ be a sequence of independent random variables, and for $i \in \mathbb{N}$, let μ_i and σ_i^2 be the expectation and variance of X_i. Let $s_n^2 = \sum_{i=1}^n \sigma_i^2$. If there exists $\delta > 0$ such that Lyapunov's condition*

$$\lim_{n \to +\infty} \frac{1}{s_n^{2+\delta}} \sum_{i=1}^{n} \mathbb{E}\left[|X_i - \mu_i|^{2+\delta} \right] = 0 \tag{53}$$

is satisfied, then the random variable $\frac{1}{s_n} \sum i = 1^n (X_i - \mu_i)$ converges in distribution to a normal random variable, as n goes to infinity:

$$\frac{1}{s_n} \sum i = 1^n (X_i - \mu_i) \xrightarrow{d} \mathcal{N}(0, 1) \tag{54}$$

Appendix D Maximum-likelihood estimation of Gumbel parameters

Given a sample of scores of sequences aligned with a single path in a protomaton, we aim at approximating their distribution with a Gumbel distribution. A regression method is possible as described in [Eddy, 1997], but the maximum likelihood estimation, introduced below, is argued to perform better.

Let us consider N independent samples $(x_i)_{1 \leqslant i \leqslant N}$ from a Gumbel distribution with parameters λ and μ. The likelihood of the samples $L(\lambda, \mu)$ is given by:

$$L(\lambda, \mu) = \prod_{i=1}^{N} \lambda \exp\left(-\lambda(x_i - \mu) - e^{-\lambda(x_i - \mu)} \right) = \lambda^N \exp\left(\sum_{i=1}^{N} -\lambda(x_i - \mu) - \sum_{i=1}^{N} e^{-\lambda(x_i - \mu)} \right) \tag{55}$$

This leads to:

$$\log L(\lambda, \mu) = N \log \lambda - \sum_{i=1}^{N} \lambda(x_i - \mu) - \sum_{i=1}^{N} e^{-\lambda(x_i - \mu)} \tag{56}$$

and:

$$\frac{\partial \log L}{\partial \mu} = N\lambda - \lambda \sum_{i=1}^{N} e^{-\lambda(x_i - \mu)} \tag{57}$$

$$\frac{\partial \log L}{\partial \lambda} = \frac{N}{\lambda} - \sum_{i=1}^{N} (x_i - \mu) - \sum_{i=1}^{N} (x_i - \mu) e^{-\lambda(x_i - \mu)} \tag{58}$$

From $\frac{\partial \log L}{\partial \mu} = 0$ we can deduce:

$$\mu = -\frac{1}{\lambda} \log \left(\frac{1}{N} \sum_{i=1}^{N} e^{-\lambda(x_i - \mu)} \right) \tag{59}$$

Finally, injecting 59 in 58 gives:

$$\frac{1}{\lambda} - \frac{1}{N} \sum_{i=1}^{N} x_i + \frac{\sum_{i=1}^{N} x_i e^{-\lambda x_i}}{\sum_{i=1}^{N} e^{-\lambda x_i}} = 0 \tag{60}$$

This equation can easily be solved thanks to the Newton algorithm for example.

Fitting censored histograms to the Gumbel distribution If we want to fit the tail of the distribution, we can set a threshold c, and censor the values x_i such that $x_i < c$. Let us denote z the number of censored values and $x_1, \dots x_{n-z}$ the non-censored samples. Then, the previous maximum-likelihood method leads to:

$$\mu = -\frac{1}{\lambda} \log \left(\frac{1}{N-z} \left(z e^{-\lambda c} + \sum_{i=1}^{N-z} e^{-\lambda(x_i - \mu)} \right) \right) \tag{61}$$

$$\frac{1}{\lambda} - \frac{1}{N-z} \sum_{i=1}^{N-z} x_i + \frac{z c e^{-\lambda c} + \sum_{i=1}^{N-z} x_i e^{-\lambda x_i}}{z e^{-\lambda c} + \sum_{i=1}^{N-z} e^{-\lambda x_i}} = 0 \tag{62}$$

Appendix E Sigmoid distribution of reversed-sequence null-model scores

Let us consider a random sequence and let us denote s_M the random variable corresponding to its log score, s'_M the variable describing the log-score of its reversed sequence. Then, $S = s_M - s'_M$ corresponds to the random variable that describes the score of the sequence with the reversed-sequence null-model. As shown in section 4, the log-scores follow a Gumble distribution. Let us denote and μ the parameters of this distribution. We are seeking the density of the distribution of S.

The law of total probability with (S, s'_M) gives:

$$f_S(a) = \int_{-\infty}^{+\infty} f_S(a|s'_M = x) f_{s'_M}(x) dx \tag{63}$$

$$\text{but } f_S(a|s'_M = x) = f_{s_M}(x - a) \text{ because } S = s_M - s'_M \tag{64}$$

$$= \lambda^2 \int_{-\infty}^{+\infty} e^{-\lambda(x-\mu)-e^{-\lambda(x-\mu)}} e^{-\lambda(x+a-\mu)-e^{-\lambda(x+a-\mu)}} dx \tag{65}$$

$$= \lambda^2 \int_{-\infty}^{+\infty} \exp\left(-2\lambda(x-\mu) - \lambda a - e^{-\lambda(x-\mu)} - e^{-\lambda(x+a-\mu)}\right) dx \tag{66}$$

$$\text{then thanks to the variable substitution } t = e^{-\lambda(x-\mu)} : \tag{67}$$

$$= \lambda^2 \int_{-\infty}^{+\infty} t^2 \exp\left(-\lambda a - t(1 + e^{-\lambda a})\right) \left(\frac{-1}{\lambda t}\right) dt \tag{68}$$

$$\text{and with integration by parts:} \tag{69}$$

$$= \frac{\lambda e^{-\lambda a}}{(1 + e^{-\lambda a})^2} = \frac{\lambda e^{\lambda a}}{(1 + e^{\lambda a})^2} \tag{70}$$

$$\tag{71}$$

Appendix F Maximum likelihood estimation of a sigmoid distribution

Let us consider N independent samples $(x_i)_{1 \leqslant i \leqslant N}$ from a sigmoid distribution of parameter λ. The likelihood of the samples $L(\lambda)$ is given by:

$$L(\lambda) = \prod_{i=1}^{N} \frac{\lambda e^{\lambda x_i}}{(1 + e^{\lambda x_i})^2} \tag{72}$$

This leads to:

$$\log L(\lambda) = N\lambda + \lambda \sum_{i=1}^{N} x_i - 2 \sum_{i=1}^{N} \log(1 + e^{\lambda x_i}) \tag{73}$$

and:

$$\frac{d \log L(\lambda)}{d\lambda} = \frac{N}{\lambda} + \sum_{i=1}^{N} x_i - 2 \sum_{i=1}^{N} \frac{x_i e^{\lambda x_i}}{1 + e^{\lambda x_i}} \tag{74}$$

We seek λ that maximizes $L(\lambda)$, i.e. such that $\frac{d \log L(\lambda)}{d\lambda} = 0$. We can use the Newton method to find the solution.